グラフィクス

解析マニュアル［第3巻］

井上智博 著

PC-9801
PC-9801E
PC-9801F
PC-9801M

SHUWA
SYSTEM
TRADING
CO.,LTD.

ISBN4-87966-029-9 C0000 ¥3200E

PC-9801/E/F/M

GRAPHICS ANALYTICAL MANUAL

秀和システム SHUWA SYSTEM TRADING CO.,LTD.

まえがき

本書はＰＣ－９８００シリーズのもつグラフィクス機能を，ハードウェア・ソフトウェアの両面から解析した結果をまとめたものです．

ＰＣ－９８００シリーズのハードウェアの大きな特徴は，グラフィックの表示，および描画にＧＤＣ(Graphic Display Controller)と呼ばれる専用ＬＳＩ(μＰＤ７２２０）を採用したことです．ＧＤＣは，表示に関して，画面分割，描画に関しては，直線，四辺形，円弧の描画，グラフィック文字の拡大描画など，表示，描画とも優れた機能を持っていますが，標準となるソフトウェアであるN88－ＢＡＳＩＣが，これらの機能を充分に生かしていないのは残念なことです．そこで本書では，ＰＣ－９８００シリーズのハードウェアの機能を最大限に引き出すことを１つの主題としました．グラフィックＶＲＡＭ，パレットレジスタなどグラフィクス関連ハードウェアとともに第１章で解説します．

ＰＣ－９８００シリーズのソフトウェア自体はよく整理されています．ＰＣ－９８００シリーズではN88－ＢＡＳＩＣを支えるグラフィクス機能をＬＩＯ(Logical Input Output)および，ＢＩＯＳ(Basic Input Output Subroutine)と呼ばれる２層のサブルーチン群としてまとめられており，それぞれソフトウェアインタラプトを用いて呼び出すことができるようになっています．これらは，ＣＰ／M，ＭＳ－ＤＯＳからも利用できます．ハードウェアにより近いＢＩＯＳについては第２章で，ＢＡＳＩＣよりのＬＩＯについては第３章で解説します．

N88－ＢＡＳＩＣのグラフィクス関連コマンドについてはフローチャートを掲載しＢＡＳＩＣインタプリタの仕組みを理解できるように努めました．特に，新しいコマンドを作ろうとする場合には非常に参考となるでしょう．

第５章には解析結果を生かした応用プログラムを集めました．５．０はＧＤＣの機能を生かしたＢＡＳＩＣの拡張コマンドを，５．１～５．３は３次元グラフィクスへの応用例です．５．１ではN88－ＢＡＳＩＣから使える３次元拡張コマンドを，５．２ではＭＳ－ＤＯＳとＦＯＲＴＲＡＮを用いた応用例，５．３ではＣＰ／Ｍ－８６上でのアセンブラによる応用例です。

また，本書の執筆にあたっては伊藤貢司氏，小松伸行氏，豊嶋敬一氏の各氏に多大な協力を仰ぎました．伊藤氏には５．２の応用例の，小松，豊島両氏には，５．３の応用例の作成，解説を賜りました．これらのプログラムによって本書の深みがさらに増し，価値が高められましたことを，ここに感謝致します．

読者諸兄が本書を通してＰＣ－９８００シリーズの再発見の喜びを味わうことができれば筆者にとって望外の幸せです．

１９８４年　１２月

井上智博

目　次

【サンプルプログラム】

本書の表記法

機能	そのルーチンの機能を簡単に説明します．
割込	ソフトウェアインタラプトによる呼び出し方法を示します．
絶対番地	そのルーチンのエントリ・アドレスを示します．
入力条件	そのルーチンをコールする前に設定すべきレジスタ，引数などを示します．
引数	入力条件の中で特に引数テーブルと呼べるものが存在する場合，入力条件として区別して示します．
出力条件	リターンしたときにレジスタ，メモリに意味ある値が格納される場合，それを示します．
破壊	ルーチンをコールすることによって内容が破壊されるレジスタを示します．
解説	ルーチンの機能について詳細に説明します．
処理	ルーチンの内部で行われている処理について説明します．
サンプル	そのルーチンを使用したプログラムの例を示します．

注意

本書では，参考のため全ルーチンについてエントリ・アドレス（絶対番地）を示しましたが，絶対番地は機種の相違，ＲＯＭのバージョンになって異なることがあります．本書に示したものはＰＣ－９８０１Ｆの場合です．ＢＩＯＳ，ＬＩＯを使用する場合は必ずソフトウエアインタラプトを用いてください．また，それ以外の内部サブルーチンを利用する場合はそのルーチンを直接コールせず，自分のプログラムにコピーして使われることをお勧めします．

解析に使用したシステム

・ＰＣ－９８０１Ｆ２　　シリアルナンバ　　４４００４９６９Ａ

・ＰＣ－８８５３Ｎ　　１４インチ・カラー専用高解像度ディスプレイ

・ＰＣ－ＰＲ２０１　　日本語シリアルプリンタ

参考文献

1．NEC，PC－9801F USER'S MANUAL.

2．NEC，PC－9801F BASIC REFERENCE MANUAL.

3．NEC，μPD7220 GDCユーザーズ・マニュアル.

4．川村清，PC－9801解析マニュアル第0巻，
秀和システムトレーディング株式会社，１９８３年6月.

5．川村清，PC－9801／E／F N88－BASIC－86解析マニュアル
第1巻　秀和システムトレーディング株式会社，１９８４年7月.

6．山口富士夫，コンピュータディスプレイによる図形処理工学，
日刊工業新聞社，１９８１年6月.

7．伊藤貢司，"三次元グラフィックスシステムの開発"
ＰＩＸＥＬ，図形処理情報センター，１９８４年3月号.

8．高澤嘉光，"ナノピコ教室解答，網目づくり"
ｂｉｔ，１９８２年7月号，ｐｐ．１０１～１０３．

第０章

グラフィクスの概要

解析に使用したシステム

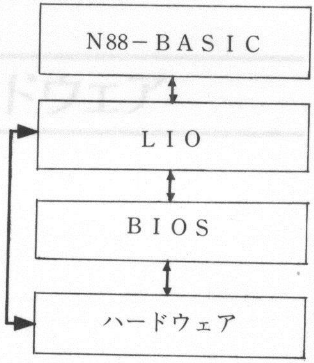
・PC－9801F2　　シリアルナンバ　　44004969A
・PC－8853N　　14インチ・カラー専用高解像度ディスプレイ
・PC－PR201　　日本語プリンタ

参考文献

1．NEC，PC－9801F USER'S MANUAL．
2．NEC，PC－9801F BASIC REFERENCE MANUAL．
3．NEC，μPD7220 GDCユーザーズ・マニュアル．
4．川村清，PC－9801解析マニュアル第0巻，
　　秀和システムトレーディング株式会社，1983年6月．
5．川村清，PC－9801/E/F N88－BASIC－86解析マニュアル
　　第1巻　　秀和システムトレーディング株式会社，1984年7月．
6．山口富士夫，コンピュータディスプレイによる図形処理工学，
　　日刊工業新聞社，1981年6月．
7．伊藤貴司，"三次元グラフィックスシステムの開発"
　　PIXEL，図形処理情報センター，1984年3月号．
8．高澤嘉光，"ナノピコ教室解答，譜面づくり"
　　bit，1982年7月号，pp．101～103．

ＰＣ９８０１は、ＰＣ８８０１と同じN_{88}－ＢＡＳＩＣを搭載し、ＣＩＲＣＬＥ、ＰＡＩＮＴなどの描画機能、ワールド座標系など高度なグラフィクスを実現しています。

ＰＣ９８００シリーズのソフトウェアは、ＢＩＯＳ(Basic Input Output Subroutine)、ＬＩＯ(Logical Input Output)など、階層的整理されており、機械語でプログラムを作る際、ＲＯＭ内のサブルーチンを利用しやすくなっています。右図にＰＣ９８０１のソフトウェアの階層構造を示します。ＢＡＳＩＣインタプリタとハードウェアは、２層のサブルーチン群によって切り離されているわけです。ＬＩＯはＢＡＳＩＣに近く、ＢＩＯＳはハードウェアに密接に結びついています。

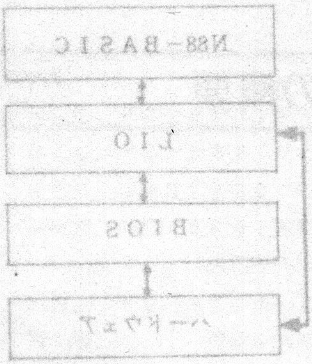


次に各階層の特徴と主な処理を挙げます。

	特　徴	主な処理
Ｎ８８－ＢＡＳＩＣ	・ワールド座標で座標を指定 ・相対座標が使える ・ビューポートが使える	・ステートメントの解釈 ・相対座標から絶対座標への変換 ・ワールド座標からスクリーン座標への変換
ＬＩＯ	・スクリーン座標で座標を指定 ・ビューポートが使える	・ビューポートによるクリッピング ・複合機能を単一機能に分解
ＢＩＯＳ	・スクリーン座標で座標を指定	・スクリーン座標からアドレスへの変換 ・ＧＤＣコマンドの作成
ハードウエア	・アドレスで座標を指定	

ＰＣ９８０１は、ＰＣ８８０１と同じＮ88－ＢＡＳＩＣを搭載し、ＣＩＲＣＬＥ、ＰＡＩＮＴなどの描画機能、ワールド座標系など高度なグラフィクスを実現しています。

ＰＣ９８００シリーズのソフトウェアは、ＢＩＯＳ(Basic Input Output Subroutine)、ＬＩＯ(Logical Input Output)など、階層的整理されており、機械語でプログラムを作る際、ＲＯＭ内のサブルーチンを利用しやすくなっています。右図にＰＣ９８０１のソフトウェアの階層構造を示します。ＢＡＳＩＣインタプリタとハードウェアは、２層のサブルーチン群によって切り離されているわけです。ＬＩＯはＢＡＳＩＣに近く、ＢＩＯＳはハードウェアに密接に結びついています。



次に各階層の特徴と主な処理を挙げます。

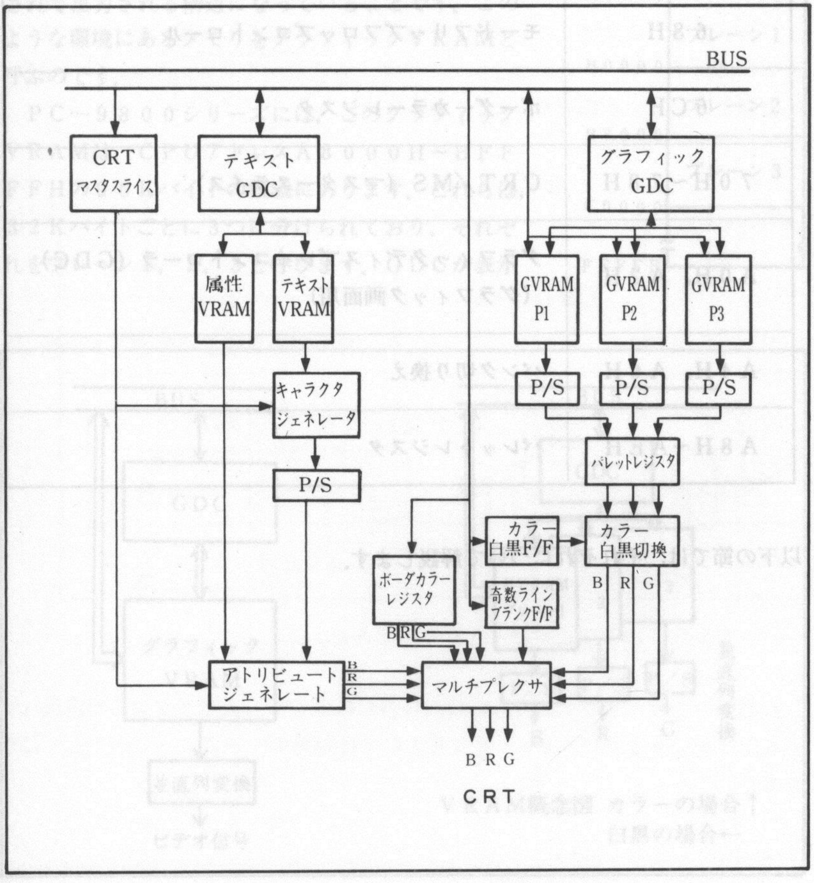
	特　　徴	主な処理
N88－BASIC	・ワールド座標で座標を指定 ・相対座標が使える ・ビューポートが使える	・ステートメントの解釈 ・相対座標から絶対座標への変換 ・ワールド座標からスクリーン座標への変換
LIO	・スクリーン座標で座標を指定 ・ビューポートが使える	・ビューポートによるクリッピング ・複合機能を単一機能に分解
BIOS	・スクリーン座標で座標を指定	・スクリーン座標からアドレスへの変換 ・GDCコマンドの作成
ハードウェア	・アドレスで座標を指定	

第1章

グラフィクスのハードウェア

1．0　ハードウェアの概要

ＰＣ－９８００シリーズのグラフィクスは広大なグラフィックメモリと多くのＬＳＩで成り立っています．ＰＣ－９８００シリーズのＣＲＴ関係のハードウェア概念図を次に示します．ＰＣ－９８００シリーズはＰＣ－８８００シリーズと同じくグラフィック画面とテキスト画面を分離し，グラフィック画面は３２Ｋバイトのプレーンを３枚持ち，６４０×４００の解像度，８色カラーを実現しています.テキスト画面は２枚あり，それぞれは，８Ｋバイトのテキストエリアと４Ｋバイトのアトリビュートエリアからなっています．



ＬＳＩおよびその他のハードウェアの多くは，Ｉ／Ｏ空間に割りつけられ，ソフトウェアからの機能設定，変更を可能にしています．各ハードウェアのＩ／Ｏアドレスの一覧表を次に示します．

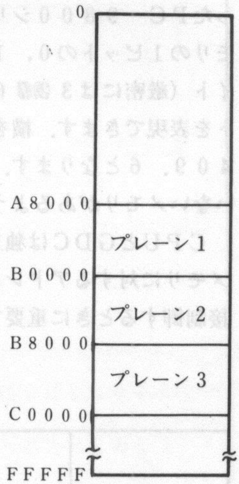
ＩＯアドレス一覧表

ＩＯアドレス	ハードウエア
６０Ｈ，６２Ｈ	グラフィックディスプレイコントローラ（ＧＤＣ）（テキスト画面用）
６８Ｈ	モードフリップフロップコントロール
６ＣＨ	ボーダーカラーレジスタ
７０Ｈ～７８Ｈ	ＣＲＴ／ＭＳ（マスタースライス）
Ａ０Ｈ，Ａ２Ｈ	グラフィックディスプレイコントローラ（ＧＤＣ）（グラフィック画面用）
Ａ４Ｈ，Ａ６Ｈ	バンク切り換え
Ａ８Ｈ～ＡＥＨ	パレットレジスタ

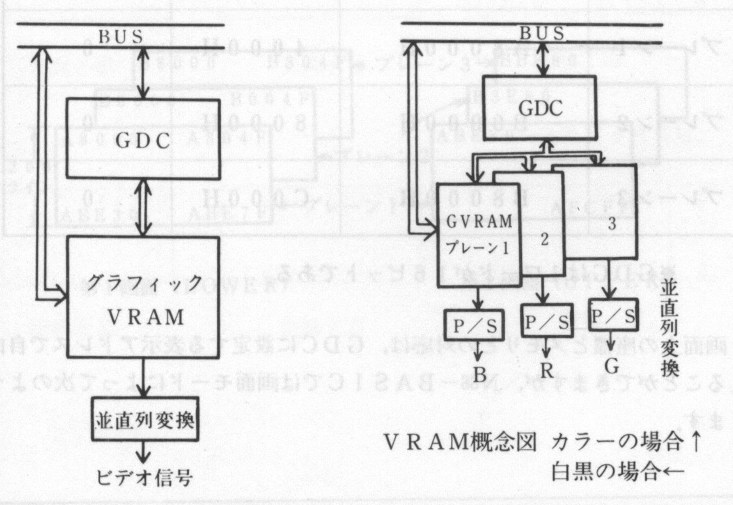
以下の節では，それぞれについて解説します．

1．1 グラフィック VRAM（ビデオRAM）

グラフィックVRAMとは，画像情報を画面上の1ドットをメモリの1ビットに対応させて蓄えておくメモリのことで，フレームバッファとも呼ばれます．このメモリが普通のメモリと異なるのは，CPUのバスに接続されているだけでなく，GDC（グラフィックディスプレイコントローラ）のバスにも接続されており，CPUと独立にGDCがアドレスを与えて内容を読み出し，それがビデオ信号に変換されて出力される構造になっていることです．このような環境にあるメモリをグラフィックVRAMと呼ぶのです．



PC—9800シリーズには，このグラフィックVRAMは，CPUアドレスA8000H～BFFFFHの96Kバイトの領域にあります．これらは，32Kバイトごとに3つに分けられており，それぞれをプレーン1，2，3と呼びます．GDCが表示



VRAM概念図 カラーの場合↑
白黒の場合←

のためにグラフィックVRAMをアクセスするときには，この3つのプレーンは同じアドレス空間にあるように作られており，同時に3つのプレーンから内容が読み出されます．このそれぞれの出力をカラーのG，R，Bに対応させることによって8色のカラーを実現できるわけです．

さて，このように1プレーンあたり32Kバイトのグラフィック VRAMを持ったPC−9800シリーズは，どれだけの解像度を実現できるでしょうか．メモリの1ビットの0，1が画面上の1ドットの明暗を表わしますから，32Kバイト（厳密には32768バイト）では，32768×8＝262144のドットを表現できます．横を640ドットとしますと，縦は262144／640＝409．6となります．実際は，最高640×400ですからほんの少し使っていないメモリがあるようです．

CPUとGDCは独立にバスを持っておりさらにGDCは描画時と表示時ではメモリに対するアドレスが異なっています．ここに示すアドレスは，GDCを直接制御するときに重要です．

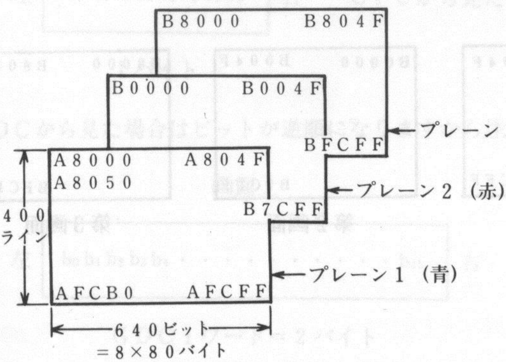
各プレーンの先頭アドレス

	CPUアドレス	GDCアドレス※	
		描画時	表示時
プレーン1	A8000H	4000H	0
プレーン2	B0000H	8000H	0
プレーン3	B8000H	C000H	0

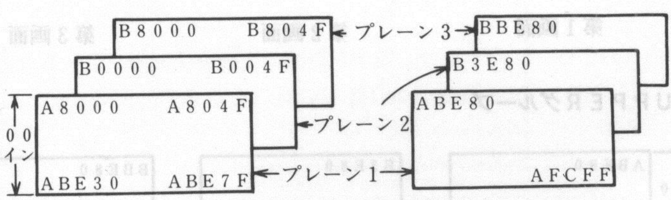
※GDCは1ワードが16ビットである．

画面上の座標とメモリとの対応は，GDCに設定する表示アドレスで自由に変えることができますが，N88−BASICでは画面モードによって次のようになります．

①６４０×４００カラーモード

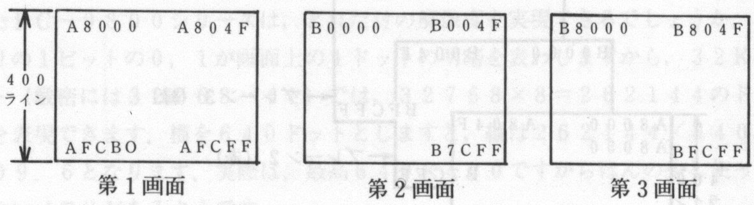


②６４０×２００カラーモード



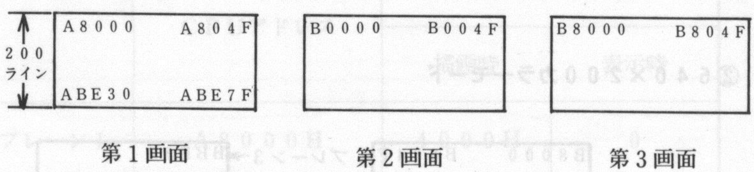
第１画面（ＬＯＷＥＲ）　　第２画面（ＵＰＰＥＲ）

③６４０×４００白黒モード

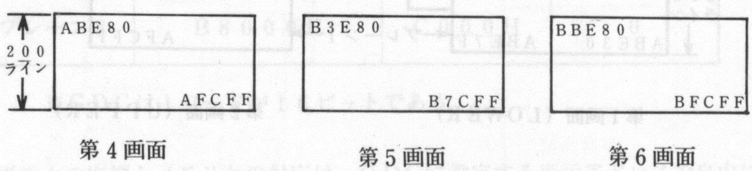


④６４０×２００白黒モード

ＬＯＷＥＲグループ



ＵＰＰＥＲグループ



1バイト内での各ビットと画面上のドットとの対応を次に示します．

画面上

左　| $b_7\,b_6\,b_5\,b_4\,b_3\,b_2\,b_1\,b_0$ |　右　　ＣＰＵから見た場合

1バイト

ＧＤＣから見た場合はビットが逆順になりますから注意して下さい．

画面上

左　| $b_0\,b_1\,b_2\,b_3\,b_4$ ・・・・・・・・・・・・ b_{15} |　右　　ＧＤＣから見た場合

ＧＤＣ1ワード＝2バイト

ＰＣ－９８０１Ｅ／Ｆ／Ｍでは，グラフィックＲＡＭがさらに増やされ，２倍の１９２Ｋバイトとなっています．これはＰＣ－９８０１と同じアドレスにもう１組９６Ｋバイトのメモリが置かれており，これらをバンク切り換えで使用するものです．

★漢字ＲＯＭの実装を調べる方法

ＰＣ－９８０１Ｆ／Ｍでは，漢字ＲＯＭは標準実装となっていますが，ＰＣ－９８０１／Ｅではオプションです．わかりやすさの点から，日本語表示は望ましいのですが，ハードウェアによっては漢字ＲＯＭが実装されていないことも考えられ，広く使われるソフトウェアを作る上で問題となります．そこでプログラム中で，自動的に漢字ＲＯＭの実装をチェックし，ない場合はカタカナ，英文等でメッセージを出すという方法が考えられます．ここでは，その漢字ＲＯＭの有無のチェック方法を紹介しましょう．

日本語表示のためには，実は漢字ＲＯＭだけでなく，テキスト用ビデオＲＡＭの増設が必要なのです．漢字ＲＯＭボードには，漢字ＲＯＭとビデオＲＡＭが載っているのです．テキスト用ビデオＲＡＭは，セグメントＡ０００Ｈからにありますが，ＡＳＣＩＩコードだけの表示には1バイトですむため，偶数アドレスだけを使います．つまり，漢字ＲＯＭボードの実装されていない機種は奇数アドレスにメモリが実装されていないのです．ですから，このメモリをチェックすればいいことがわかります．

具体的には実アドレス　Ａ０００１Ｈに００とＦＦＨを書き込んで読み出せれば実装がわかります．

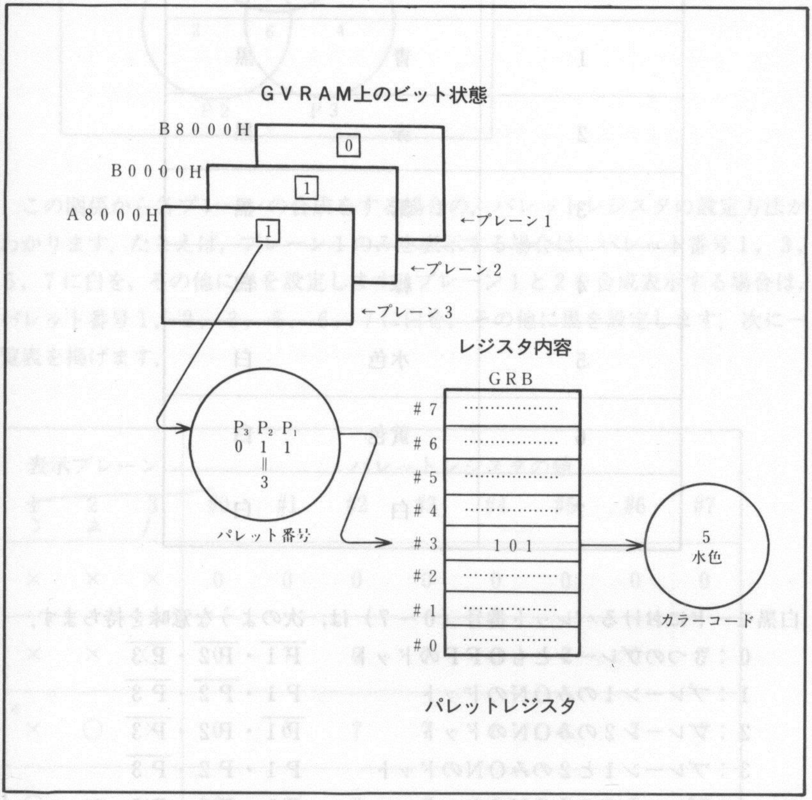
プログラム

このＢＡＳＩＣプログラムは，漢字ＲＯＭの実装をチェックし，あれば“Kanji ROM Ari.”，なければ，“Kanji ROM Nashi.” とＣＲＴに出力します．

```
1 '*****************************************
2 '*   KANJI ROM GA ARU KA NAIKA O SHIRABERU   *
3 '*****************************************
10 DEF SEG=&HA000          :'ﾃｷｽﾄ VRAM ﾉ ｴﾘｱ ﾆ ｾｯﾄ
20 ' 1 ﾊﾞﾝﾁﾆ 0 ﾄ &hFF ｦ ｶｷｺﾐ、 ﾖﾐﾀﾞｽ
30 FOR I=&HFF TO 0 STEP -&HFF
40    POKE 1,I
50    IF PEEK(1)<>I THEN 80
60 NEXT I
70 PRINT "KANJI ROM ARI.":END
80 PRINT "KANJI ROM NASHI.":END
```

1．2 パレットレジスタ

パレット番号から，カラーコードへの対応を定義するレジスタです．ＰＣ－９８００シリーズでは，８色のカラーを使用できますが，それは，前述の３つのプレーン上に書かれているビットが，そのまま，光の３原色に対応するのではなく，３つのプレーン上に書かれているビットから決まる値（パレット番号と呼ぶ）と，実際の色との対応を１つひとつ定義することができます．それを行なうのがパレットレジスタです．



白黒モードのときには，また，パレットレジスタは，白黒画面（プレーン１～３）を合成する働きをします．動作はカラーモードのときとほとんど同じです．プレーン１～３からパレット番号に対応するパレットレジスタの内容が出力されます．ただし，出力されるカラーコードと実際の色との対応は，次のようになりま

す．普通，白黒モードでは，カラーコードは，０と７のみを使います．

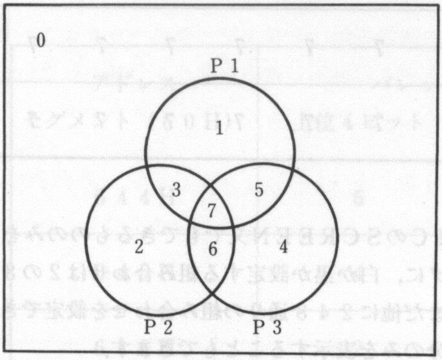
カラーコードと実際の色との対応

カラーコード	実際の色	
	カラーモード	白黒モード
0	黒	黒
1	青	黒
2	赤	黒
3	紫	黒
4	緑	白
5	水色	白
6	黄色	白
7	白	白

白黒モードにおけるパレット番号（０～７）は，次のような意味を持ちます．

０：３つのプレーンともＯＦＦのドット　$\overline{P1} \cdot \overline{P2} \cdot \overline{P3}$
１：プレーン１のみＯＮのドット　$P1 \cdot \overline{P2} \cdot \overline{P3}$
２：プレーン２のみＯＮのドット　$\overline{P1} \cdot P2 \cdot \overline{P3}$
３：プレーン１と２のみＯＮのドット　$P1 \cdot P2 \cdot \overline{P3}$
４：プレーン３のみＯＮのドット　$\overline{P1} \cdot \overline{P2} \cdot P3$
５：プレーン１と３のみＯＮのドット　$P1 \cdot \overline{P2} \cdot P3$
６：プレーン２と３のみＯＮのドット　$\overline{P1} \cdot P2 \cdot P3$
７：３つのプレーンともＯＮのドット　$P1 \cdot P2 \cdot P3$

図示すると次のようになります．



P1：プレーン1
P2：プレーン2
P3：プレーン3

0～7：パレット番号

この関係から各プレーンの合成をする場合の，パレットレジスタの設定方法がわかります．たとえば，プレーン1のみを表示する場合は，パレット番号1，3，5，7に白を，その他に黒を設定します．プレーン1と2を合成表示する場合は，パレット番号1，2，3，5，6，7に白を，その他に黒を設定します．次に一覧表を掲げます．

表示プレーン			パレットレジスタの値							
1	2	3	#0	#1	#2	#3	#4	#5	#6	#7
×	×	×	0	0	0	0	0	0	0	0
×	×	○	0	7	0	7	0	7	0	7
×	○	×	0	0	7	7	0	0	7	7
○	×	×	0	0	0	0	7	7	7	7
×	○	○	0	7	7	7	0	7	7	7

○	×	○	0	7	0	7	7	7	7	7
○	○	×	0	0	7	7	7	7	7	7
○	○	○	0	7	7	7	7	7	7	7

この表には，N88－ＢＡＳＩＣのＳＣＲＥＥＮ文でもできるもののみを示しました．８個のパレットレジスタに，白か黒か設定する組み合わせは2の８乗＝２５６通りあります．つまり，まだ他に２４８通りの組み合わせを設定できます．プレーン１と2の重なった部分のみを表示することもできます．

パレットレジスタは，Ｉ／Ｏ空間の，０Ａ８Ｈ，０ＡＡＨ，０ＡＣＨ，０ＡＥＨにあります．各アドレスは，上下各4ビットが１つのパレット番号に対応しています．パレットレジスタには，書き込みのみできます．

アドレス	パレット番号 上位4ビット	 下位4ビット
Ａ８Ｈ	3	7
ＡＡＨ	1	5
ＡＣＨ	2	6
ＡＥＨ	0	4

１つのアドレスに2つのパレットレジスタが割り当てられているので，一方を変えようとすると他方も同時に変わってしまいます．一方のみ変えるためには，前に書き込んだ値を覚えておいて使うしかありません．ＢＡＳＩＣあるいはグラフィックＬＩＯを使っている場合は，ＬＩＯのワークエリアに値が保存されてい

ますので，それを使うことができます．ただし，パレットレジスタの並びが，アウトプットアドレスの場合とは異なるので，注意して下さい．

アドレス セグメント（６０Ｈ）	パレット番号 上位４ビット	 下位４ビット
６４４Ｈ	６	７
６４５Ｈ	４	５
６４６Ｈ	２	３
６４７Ｈ	０	１

●機械語による直接制御

例　パレットレジスタ３にカラーコード２を，パレットレジスタ７にカラーコード５を設定します．

```
MOV   AL, 25H
OUT   0A8H, AL
```

●ＢＩＯＳによる制御

２．４のカラーパレットの設定に相当します．このルーチンでは，４バイトの保存領域の内容をパレットレジスタに転送するだけです．保存領域は，上記のＬＩＯのワークエリアを利用するのが便利です．パレットレジスタ全体を初期化するときなどは，このＢＩＯＳを使うのがいいでしょう．

●ＬＩＯによる制御

カラーモードか白黒モードかで，使用するＬＩＯルーチンが異なります．カラーモードの場合は，３．５のＧＣＯＬＯＲ２が相当します．ＢＩＯＳと違い，パレットレジスタの１つだけを変更することができます．白黒モードの場合は，３．２のＧＳＣＲＥＥＮのディスプレイ画面のパラメータが相当します．

I．3　GDC（グラフィックディスプレイコントローラ）

GDCは表示，および描画の機能を持つLSIで，PC－9800シリーズではテキスト画面用と，グラフィック画面用とにそれぞれ1つ使われています．

□I／Oアドレス

GDCは，ステータスレジスタ，データレジスタ，パラメータレジスタ，コマンドレジスタの4つのレジスタを持っています．各レジスタのI／Oアドレスを次に示します．

	R／W	テキスト用	グラフィック用
ステータスレジスタ	READ	60H	0A0H
データレジスタ	READ	62H	0A2H
パラメータデジスタ	WRITE	60H	0A0H
コマンドレジスタ	WRITE	62H	0A2H

ステータスレジスタ，データレジスタは読み出し専用で，パラメータレジスタ，コマンドレジスタは書き込み専用です．

□GDCのコマンド

GDCの制御は，コマンドレジスタにコマンドを書き込むことによって行ないます．コマンドによってはさらにパラメータを必要とするものがあり，その場合はコマンドに続いてパラメータレジスタにパラメータを書き込みます．GDCには次に示す21種類のコマンドがあります．コマンドは大きくわけて，動作制御コマンド，表示制御コマンド，描画制御コマンド，映像メモリ制御コマンドがありますが，テキスト用GDCは動作制御，および表示制御コマンドのみを，グラフィック用GDCはさらに描画制御コマンドを使っています．

□コマンド一覧

動作制御コマンド

<table>
<tr><th rowspan="2">コマンド</th><th rowspan="2">動作内容</th><th rowspan="2">C/P</th><th colspan="8">コード</th></tr>
<tr><th>DB7</th><th>DB6</th><th>DB5</th><th>DB4</th><th>DB3</th><th>DB2</th><th>DB1</th><th>DB0</th></tr>
<tr><td>RESET</td><td>初期化動作</td><td>C</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td rowspan="9">SYNC</td><td rowspan="9">動作モード，同期信号波形の定義</td><td>C</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>DE</td></tr>
<tr><td>P1</td><td>0</td><td>0</td><td>CHR</td><td>F</td><td>I</td><td>D</td><td>G</td><td>S</td></tr>
<tr><td>P2</td><td colspan="8">←C/R→</td></tr>
<tr><td>P3</td><td colspan="3">←VS_L→</td><td colspan="5">←HS→</td></tr>
<tr><td>P4</td><td colspan="6">←HFP→</td><td colspan="2">←VS_H→</td></tr>
<tr><td>P5</td><td>0</td><td>0</td><td colspan="6">←HBP→</td></tr>
<tr><td>P6</td><td>0</td><td>0</td><td colspan="6">←VFP→</td></tr>
<tr><td>P7</td><td colspan="8">←L/F_L→</td></tr>
<tr><td>P8</td><td colspan="6">←VBP→</td><td colspan="2">←L/F_H→</td></tr>
<tr><td>MASTER / SLAVE</td><td>マスタ動作，スレーブ動作の選択</td><td>C</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>M</td></tr>
</table>

表示制御コマンド

<table>
<tr><th rowspan="2">コマンド</th><th rowspan="2">動作内容</th><th rowspan="2">C/P</th><th colspan="8">コード</th></tr>
<tr><th>DB7</th><th>DB6</th><th>DB5</th><th>DB4</th><th>DB3</th><th>DB2</th><th>DB1</th><th>DB0</th></tr>
<tr><td rowspan="2">START(1)</td><td rowspan="2">表示の開始の指示</td><td rowspan="2">C</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>STOP</td><td>表示の停止の指示</td><td>C</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td rowspan="2">ZOOM</td><td rowspan="2">拡大表示係数，拡大描画係数の設定</td><td>C</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>P</td><td colspan="4">←ZR→</td><td colspan="4">←ZW→</td></tr>
<tr><td rowspan="4">SCROLL</td><td rowspan="4">表示開始アドレス，表示領域の設定</td><td>C</td><td>0</td><td>1</td><td>1</td><td>1</td><td colspan="4">←RA→</td></tr>
<tr><td>P1</td><td colspan="8">内蔵データRAMのフォーマットに従ってパラメータを設定</td></tr>
<tr><td>⋮</td><td colspan="8"></td></tr>
<tr><td>P16</td><td colspan="8"></td></tr>
<tr><td rowspan="4">CSRFORM</td><td rowspan="4">文字表示時のカーサ形状等の設定</td><td>C</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>P1</td><td>CS</td><td>0</td><td>0</td><td colspan="5">←L/R→</td></tr>
<tr><td>P2</td><td colspan="2">←BL_L→</td><td>BD</td><td colspan="5">←CST→</td></tr>
<tr><td>P3</td><td colspan="5">←CFI→</td><td colspan="3">←BL_H→</td></tr>
<tr><td rowspan="2">PITCH</td><td rowspan="2">映像メモリ水平方向ワード数の設定</td><td>C</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>P1</td><td colspan="8">←P→</td></tr>
<tr><td rowspan="4">LPEN(2)</td><td rowspan="4">ライトペン・アドレスの読み出し指示</td><td>C</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td></td><td colspan="8">←LAD_L→</td></tr>
<tr><td></td><td colspan="8">←LAD_M→</td></tr>
<tr><td></td><td>×</td><td>×</td><td>×</td><td>×</td><td>×</td><td>×</td><td colspan="2">←LAD_H→</td></tr>
</table>

注(1)　コマンド・コード は6BHまたは0DHの8ビットを使用します。

注(2)　LPENコマンドの発行後LAD_L, LAD_M, LAD_H, の順にCPUが読み出すことができます。

描画制御コマンド

コマンド	動作内容		C/P	コード							
				DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
VECTW	描画に必要な各種パラメータの設定		C	0	1	0	0	1	1	0	0
			P1	SL	R	C	T	L	← DIR →		
			P2	← DC_L →							
			P3	×	DGD	← DC_H →					
			P4	← D_L →							
			P5	×	×	← D_H →					
			P6	← $D2_L$ →							
			P7	×	×	← $D2_H$ →					
			P8	← $D1_L$ →							
			P9	×	×	← $D1_H$ →					
			P10	← DM_L →							
			P11	×	×	← DM_H →					
VECTE	直線,四辺形,円弧,1ドット描画の実行の指示		C	0	1	1	0	1	1	0	0
TEXTW	グラフィック・テキスト・コード設定		C	0	1	1	1	1	← RA →		
			P1	← TX8, PTN_L →							
			P2	← TX7, PTN_H →							
			P3	← TX6 →							
			P4	← TX5 →							
			P5	← TX4 →							
			P6	← TX3 →							
			P7	← TX2 →							
			P8	← TX1 →							
TEXTE	グラフィック・テキスト描画実行指示		C	0	1	1	0	1	0	0	0
CSRW	描画アドレスの設定	文字モード	C	0	1	0	0	1	0	0	1
			P1	← EAD_L →							
			P2	0	0	0	← EAD_H →				
		文字／グラフィック混在モードで文字表示／描画	C	0	1	0	0	1	0	0	1
			P1	← EAD_L →							
			P2	← EAD_H →							
		文字／グラフィック混在モードでグラフィック表示／描画	C	0	1	0	0	1	0	0	1
			P1	← EAD_L →							
			P2	← EAD_H →							
			P3	← dAD →				0	0	0	0
		グラフィック・モード	C	0	1	0	0	1	0	0	1
			P1	← EAD_L →							
			P2	← EAD_M →							
			P3	← dAD →				0	0	← EAD_H →	
CSRR(3)	描画アドレスの読み出し指示		C	1	1	1	0	0	0	0	0
				← EAD_L →							
				← EAD_M →							
				×	×	×	×	×	×	← EAD_H →	
				← dAD_L →							
				← dAD_H →							
MASK	マスク・レジスタ値の設定		C	0	1	0	0	1	0	1	0
			P1	← $MASK_L$ →							
			P2	← $MASK_H$ →							

注(3)　CSRRコマンドの発行後EAD_L，EAD_M，EAD_H，dAD_L，dAD_Hの順にCPUが読み出すことができます。

映像メモリ制御コマンド

コマンド	動作内容	C/P	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
WRITE	パラメータの映像メモリへの書き込み準備	C	0	0	1	←WLH→		0	←MOD→	
		P1								
READ	映像メモリ・データの読み出し指示	C	1	0	1	←WLH→		0	←MOD→	
DMAW	映像メモリへのDMA転送開始の指示	C	0	0	1	←WLH→		1	←MOD→	
DMAR	映像メモリからのDMA転送開始の指示	C	1	0	1	←WLH→		1	←MOD→	

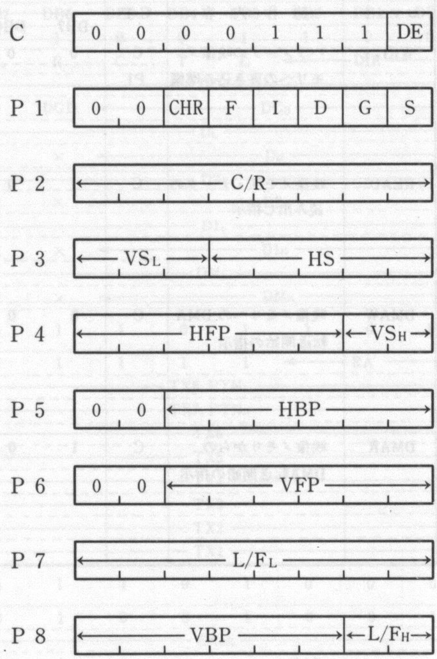
□ＧＤＣコマンドの説明とパラメータの設定法

ＧＤＣに対するコマンドおよびパラメータを設定するときは，まず，コマンドレジスタにコマンドを書き込み（ＯＵＴ　＆ＨＡ２，ＸＸ等），次にパラメータレジスタにパラメータを必要な数だけ続けて書き込みます（ＯＵＴ　＆ＨＡ０，ＸＸ等）．パラメータは常にすべて書き込む必要はありません．残りの部分は前の値が使われます．

〈表示関係コマンド〉

１．ＳＹＮＣコマンド

表示動作モード，縦，横の文字数，ライン数などの設定をします．パラメータの一覧表を右図に示します．Ｃ／Ｒを設定したときは必ずＰＩＴＣＨコマンドを続いて行なって下さい．



(1) ＤＥ　表示の開始・停止

ＤＥ	機　　能
0	表示を停止する
1	表示を開始する

パラメータを設定するときは，画面が乱れないように，通常は表示停止とする．

(2) ＣＨＲ，Ｇ　文字モード，グラフィックモード等の選択

CHR	G	モ　ー　ド
0	0	文字／グラフィック混在モード
1	0	文字モード
0	1	グラフィック・モード
1	1	設定できません

テキスト用GDC（アドレス60H）は文字モード，グラフィック用GDC（アドレスA0H）はグラフィックモードとする．

(3) F 描画タイミング

F	機　　能
0	フラッシュ描画
1	フラッシュレス描画

高速描画のときは，F＝0とする．ただし，画面はちらつく．

(4) I，S インタレース走査の有無等

I	S	機　　能
0	0	ノン・インタレース
0	1	設定できません
1	0	インタレース
1	1	インタレース・シュリンク

テキスト用GDC，グラフィック用GDCともノン・インタレースモードを使います．

(5) D メモリ・リフレッシュ

D	機　　能
0	リフレッシュ動作無し
1	リフレッシュ動作有り

テキスト用GDCではリフレッシュなし，グラフィック用GDCではリフレッシュありとします．

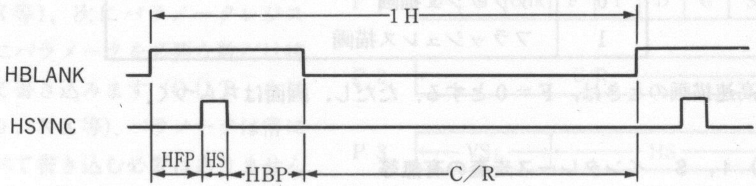
(6) C／R 1行あたりの表示文字数

C／R＝文字数－2という関係になっているので，テキスト画面で横80字の場合は，80－2＝78（16進では4EH）とします．グラフィック画面では16ドットで1文字と数え，640ドットの場合は　640／16

C／R	文字数
00000000	2
00000010	4
00000100	6
----------	--------
11111100	254
11111110	256

−２＝３８（１６進では２６Ｈ）とします．Ｃ／Ｒには偶数しか設定できません．このパラメータを設定したときは必ず後でＰＩＴＣＨコマンドを行なって下さい．

本パラメータおよび以下のパラメータによって，水平同期信号は次のように生成されます．



このように，ＨＦＰ，ＨＳ，ＨＢＰ，Ｃ／Ｒの合計が水平の１周期となります．それぞれ文字数が単位となっていますが，１文字の周期はディスプレイによって異なり，ＰＣ−９８００シリーズでは次の値となっています．

ディスプレイ	周期（テキスト用ＧＤＣ）	周期（グラフィック用ＧＤＣ）
専用高解像度	０．３８μＳＥＣ	０．７６μＳＥＣ
標準解像度	０．５６μＳＥＣ	１．１２μＳＥＣ

また，水平同期周波数は使用するディスプレイによって決まっていますから，上記パラメータの合計はそれを満たすように設定しなくてはなりません．例えば，水平同期周波数を２４．４ｋＨｚとする水平１周期は１／２４．４ｋＨｚ＝４１μＳＥＣとなりますから，１周期の文字数の合計はグラフィック用ＧＤＣの場合，４１／０．７６＝５４となります．一般にディスプレイの水平同期周波数は次のようになっています．

ディスプレイ	水平同期周波数	水平１周期
専用高解像度	２４．４ｋＨｚ	４１μＳＥＣ
標準解像度	１５．７５ｋＨｚ	６３．５μＳＥＣ

(7) ＨＳ　水平同期信号の幅

HS	文　字　数
0 0 0 0 0	1
0 0 0 0 1	2
0 0 0 1 0	3
------------	------
1 1 1 1 0	31
1 1 1 1 1	32

2文字以上を設定します。

(8) ＨＦＰ　ＣＲＴ管面右方の非表示区間

HFP	文　字　数
0 0 0 0 0 0	設定できません
0 0 0 0 0 1	2
0 0 0 0 1 0	3
------------	------
1 1 1 1 1 0	63
1 1 1 1 1 1	64

4文字以上を設定します。

(9) ＨＢＰ　ＣＲＴ管面左方の非表示区間

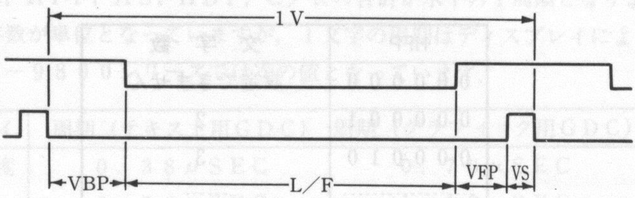
HBP	文　字　数
0 0 0 0 0 0	1
0 0 0 0 0 1	2
0 0 0 0 1 0	3
------------	------
1 1 1 1 1 0	63
1 1 1 1 1 1	64

3文字以上を設定します。

(10) ＶＳ　垂直同期信号の幅

VS_H VS_L	ライン数
0 0 0 0 0	設定できません
0 0 0 0 1	1
0 0 0 1 0	2
---------------	------
1 1 1 1 0	30
1 1 1 1 1	31

垂直同期信号は本パラメータおよび以下のパラメータによって次のように生成されます．



ＶＳ，ＶＦＰ，ＶＢＰ，Ｌ／Ｆのライン数の合計が垂直の１周期となります．ディスプレイの垂直同期周波数，水平同期周波数から合計のライン数は決まります．垂直の１周期の水平の１周期で割ったものが合計ライン数になります．例えば，垂直の１周期を１．８０５ｍＳＥＣ，水平の１周期を４１μＳＥＣとすると，ライン数＝１８０５０／４１＝４４０本となります．

	垂直同期周波数	垂直１周期	ライン数
	５５．４Ｈｚ	１．８０５ｍＳＥＣ	４４０本
	６０Ｈｚ	１．６６６ｍＳＥＣ	２６２本

(11) ＶＦＰ　ＣＲＴ管面下方の非表示区間

VFP	ライン 数
0 0 0 0 0 0	設定できません
0 0 0 0 0 1	1
0 0 0 0 1 0	2
....	......
1 1 1 1 1 0	62
1 1 1 1 1 1	63

(12) ＶＢＰ　ＣＲＴ管面上方の非表示区間

VBP	ライン 数
0 0 0 0 0 0	設定できません
0 0 0 0 0 1	1
0 0 0 0 1 0	2
....	......
1 1 1 1 1 0	62
1 1 1 1 1 1	63

(13) Ｌ／Ｆ　１画面あたりの表示ライン数

<table>
<tr><th>L/F_H</th><th>L/F_L</th><th>ライン数</th></tr>
<tr><td colspan="2">0 0 0 0 0 0 0 0 0 0</td><td>1024</td></tr>
<tr><td colspan="2">0 0 0 0 0 0 0 0 0 1</td><td>1</td></tr>
<tr><td colspan="2">0 0 0 0 0 0 0 0 1 0</td><td>2</td></tr>
<tr><td colspan="2">0 0 0 0 0 0 0 0 1 1</td><td>3</td></tr>
<tr><td colspan="2">..................</td><td>......</td></tr>
<tr><td colspan="2">1 1 1 1 1 1 1 1 1 0</td><td>1022</td></tr>
<tr><td colspan="2">1 1 1 1 1 1 1 1 1 1</td><td>1023</td></tr>
</table>

設定例

上記パラメータの設定に当たってはテキスト用ＧＤＣとグラフィック用ＧＤＣの同期をとることに注意して下さい．つまり，水平同期関係では，テキスト用の文字数が，グラフィック用の文字数の２倍となるように，垂直同期関係では，テ

キスト用のライン数と，グラフィック用のライン数が一致するように設定した下さい．次に専用高解像度ディスプレイの場合の標準的な設定例を示します．

パラメータ	テキスト用	グラフィック用
C／R	８０文字	４０文字
ＨＳ	８文字	４文字
ＨＦＰ	１０文字	５文字
ＨＢＰ	８文字	４文字
ＶＳ	８ライン	８ライン
ＶＦＰ	７ライン	７ライン
ＶＢＰ	２５ライン	２５ライン
L／F	４００ライン	４００ライン

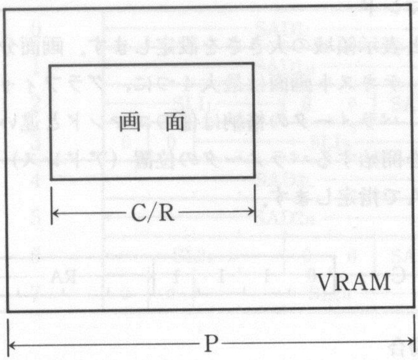
２．ＰＩＴＣＨコマンド

C	0	1	0	0	0	1	1	1
P1	←—			P				—→

ＶＲＡＭ上の１行の文字数を設定します．表示文字数と同じ場合は，ＳＹＮＣコマンドのC／Rと同じになります．C／RよりPを大きくとることによって，ＶＲＡＭ上の１部のみを画面に表示できます．

PはC／Rと違い，文字数がそのまま値となりますから注意して下さい．

P	文字数／行
0 0 0 0 0 0 0 0	0
0 0 0 0 0 0 0 1	1
0 0 0 0 0 0 1 0	2
⋮	⋮
1 1 1 1 1 1 1 0	254
1 1 1 1 1 1 1 1	255



3．STARTコマンド

表示を開始します.

C	0	1	1	0	1	0	1	1

または

C	0	0	0	0	1	1	0	1

使用例

(1) OUT　&H62，&HD（テキスト画面の場合）

(2) OUT　&HA2，&HD（グラフィック画面の場合）

4．STOPコマンド

表示を停止します.

C	0	0	0	0	1	1	0	0

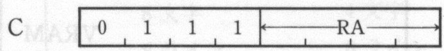
使用例

(1) OUT　&H62，&HC（テキスト画面の場合）

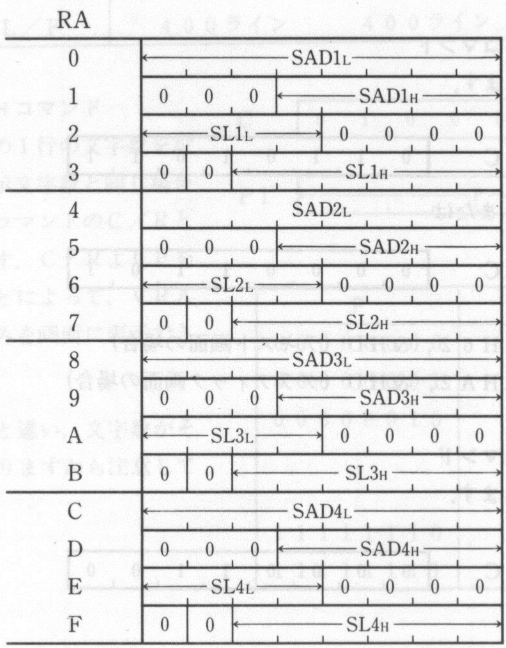
(2) OUT　&HA2，&HC（グラフィック画面の場合）

5．ＳＣＲＯＬＬコマンド

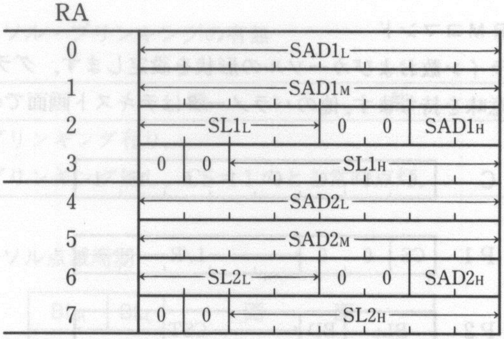
表示開始アドレスと表示領域の大きさを設定します．画面分割をするにはこのコマンドを用います．テキスト画面は最大４つに，グラフィック画面は最大２つに画面分割できます．パラメータの格納は他のコマンドと違い，任意の位置から格納できます．格納を開始するパラメータの位置（アドレス）をコマンドコードの下位４ビットのＲＡで指定します．



(1) テキスト画面の場合

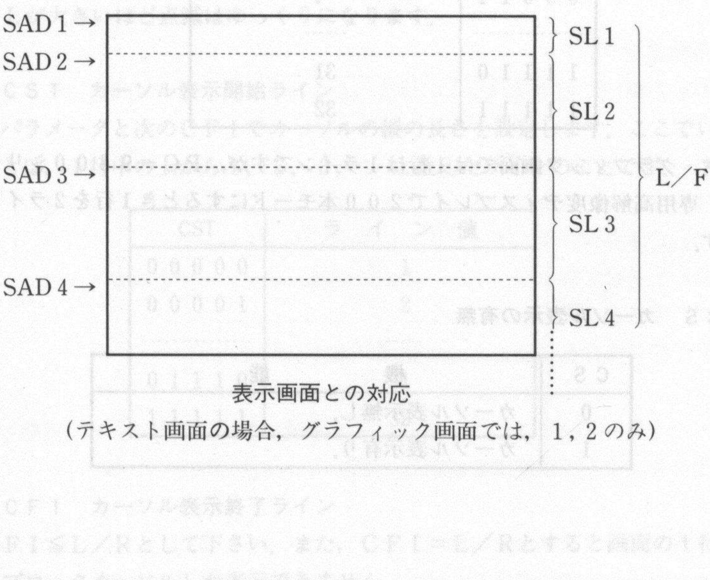


(2) グラフィック画面の場合



ＳＡＤは表示開始アドレス，ＳＬは画面分割領域の大きさを示すライン数です．ＳＡＤはＧＤＣのアドレスですので注意して下さい．テキスト第１画面は０，第２画面は８００Ｈが先頭アドレスです．

グラフィック画面の表示アドレスに関しては１．１グラフィックＶＲＡＭを参照して下さい．



表示画面との対応

（テキスト画面の場合，グラフィック画面では，１，２のみ）

6. ＣＳＲＦＯＲＭコマンド

１行中の表示ライン数およびカーソルの形状を設定します．グラフィック画面ではL／Rのみ意味を持ちます．他のパラメータはテキスト画面でのみ有効です．

C	0	1	0	0	1	0	1	1
P 1	CS	0	0	←L/R→				
P 2	BL_L		BD	←CST→				
P 3	←CFI→					←BL_H→		

(1) L／R　１行中の表示ライン数

L／R	ライン数
0 0 0 0 0	1
0 0 0 0 1	2
0 0 0 1 0	3
0 0 0 1 1	4
------------	------
1 1 1 1 0	31
1 1 1 1 1	32

通常，グラフィック画面では１行は１ラインですが，ＰＣ－９８００シリーズでは，専用高解像度ディスプレイで２００本モードにするとき１行を２ラインにします．

(2) ＣＳ　カーソル表示の有無

ＣＳ	機能
0	カーソル表示無し．
1	カーソル表示有り．

(3) ＢＤ　カーソル・ブリンキングの有無

BD	機　　　　能
0	ブリンキング有り．
1	ブリンキング無し．CS＝1のとき常時点灯．

(4) ＢＬ　カーソル点滅周期

BL_H	BL_L	画　　面
0 0 0 0 0		設定できません
0 0 0 0 1		4
0 0 0 1 0		8
0 0 0 1 1		12
0 0 1 0 0		16
0 0 1 0 1		20
‥‥‥‥		‥‥
1 1 1 1 0		120
1 1 1 1 1		124

ＢＬが大きいほど点滅はゆっくりになります．

(5) ＣＳＴ　カーソル表示開始ライン

本パラメータと次のＣＦＩでカーソルの縦の長さを設定します．ここでいうラインとは，１行中の上から１，２，３，・・と数えたラインのことです．

CST	ラ　イ　ン　値
0 0 0 0 0	1
0 0 0 0 1	2
‥‥‥‥	‥‥
0 1 1 1 0	31
1 1 1 1 1	32

(6) ＣＦＩ　カーソル表示終了ライン

ＣＦＩ≦Ｌ／Ｒとして下さい．また，ＣＦＩ＝Ｌ／Ｒとすると画面の１行目では，ブロックカーソルしか表示できません．

CFI	ラ イ ン 値
０００００	１
０００01	２
......	......
１１１１０	31
１１１１１	32

設定例

ＰＣ－９８００シリーズのテキスト画面では通常次のように設定されています．

パラメータ	専用高解像度		標準解像度	
	２５行	２０行	２５行	２０行
Ｌ／Ｒ(ライン数)	１６	２０	８	１０
ＢＬ	１２	１２	１２	１２
ＣＳＴ(ライン値)	１	１	１	１
ＣＦＩ(ライン値)	１６	２０	８	１０

〈描画関係コマンド〉

ＧＤＣを用いて描画を行なうには次のようにコマンドを与えます．

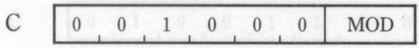
(1) ＷＲＩＴＥコマンド……ドット修正モードの設定
(2) ＴＥＸＴＷコマンド……ラインスタイル，文字パターンの設定
(3) ＣＳＲＷコマンド………描画開始アドレスの設定
(4) ＶＥＣＴＷコマンド……描画の種類，パラメータを設定
(5) ＶＥＣＴＥまたは
　　ＴＥＸＴＥコマンド……描画の実行を指示

以上のうち，(1)と(2)と(3)は常に与える必要はなく，目的に応じて省略できます．次に各コマンドの詳細を説明します．

Ｉ．ＷＲＩＴＥコマンド

ドット修正モードの設定をします．コマンドコードの下位２ビットの値によっ

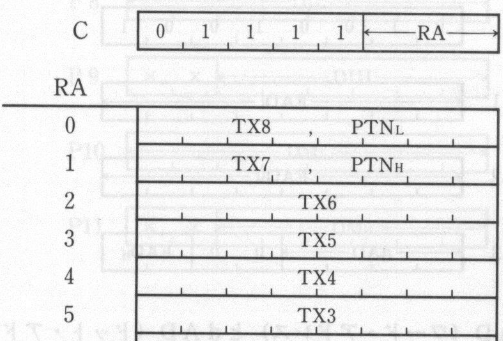
て次のように4つのモードを選択できます．1度設定すると再設定を行なうまで有効です．



MOD	機　　能
0 0	REPLACE
0 1	COMPLEMENT
1 0	CLEAR
1 1	SET

REPLACEとは，描画パターンの1，0をそのまま書き込むこと，SETとは，描画パターンのうち1のみを書き込むこと，CLEARとは，描画パターンのうち1のみを0として書き込むこと．COMPLEMENTとは，描画パターンのうちの1と対応するVRAMのドットを反転することを意味します．描画パターンはTEXTWコマンドで与えられます．

2．TEXTWコマンド



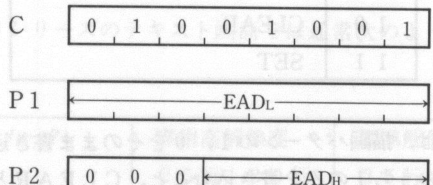
ラインスタイル，または，グラフィック文字描画時の文字パターンを設定します．このパターン記憶用の内蔵RAMは8バイトあり，全体で8×8の文字パタ

ーンを保持します．先頭の2バイトはラインスタイルと共用です．この内蔵ＲＡＭへは任意のアドレスから書き込むことができ，コマンドコードの下位3ビットでその先頭アドレスを指定します．

3．ＣＳＲＷコマンド

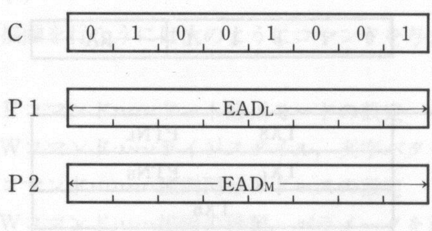
描画開始点を設定します．テキスト用ＧＤＣに対しては，カーソル位置の設定として働きます．

(1) テキスト用ＧＤＣの場合



ＥＡＤでカーソルのアドレスを指定します．アドレスはＧＤＣのアドレスであることに注意して下さい．

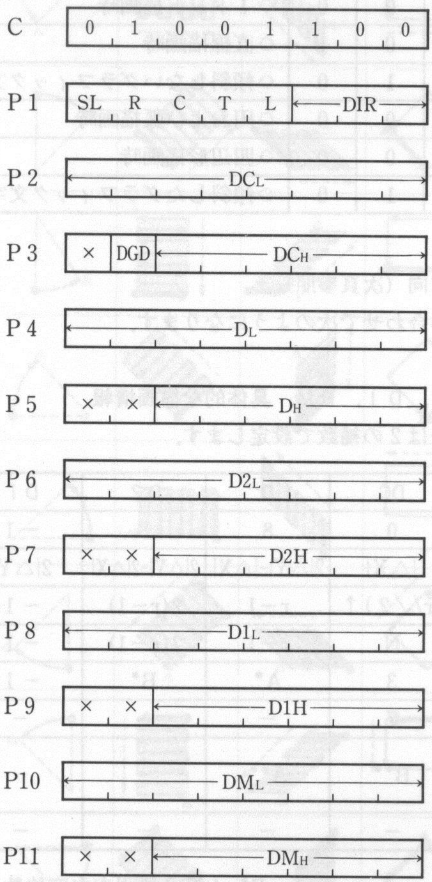
(2) グラフィック用ＧＤＣの場合



描画開始点をＥＡＤ（ワード・アドレス）とｄＡＤ（ドット・アドレス）で，指定します．ワード・アドレスとは，目的のドットが含まれているワードの（ＧＤＣから見た）アドレスであり，ドット・アドレスとは，そのワードの中の位置を示すものです．1ワード中では，1番左がビット0です．（1．1参照）

4. VECTWコマンド

直線，円弧などの描画種類，描画方向，具体的な描画情報を設定します．



(1) SL，R，C，T，L　描画種類の設定

SL	R	C	T	L	
0	0	0	0	0	○1ドット描画時
0	0	0	0	1	○直線描画時
0	0	0	1	0	○傾斜しないグラフィック文字描画時
0	0	1	0	0	○円および弧描画時
0	1	0	0	0	○四辺形描画時
1	0	0	1	0	○傾斜したグラフィック文字描画時

(2) DIR　描画方向（次頁参照）

描画種類との組み合わせで次のようになります.

(3) DC，D，D2，DI，DM　具体的な描画情報

値が負になる場合は2の補数で設定します.

	DC	D	D2	DI	DM
初期値	0	8	8	−1	−1
直線	$\|\triangle X\|$	$2\|\triangle Y\|-\|\triangle X\|$	$2\|\triangle Y\|-2\|\triangle X\|$	$2\|\triangle Y\|$	−
円	$(r/\sqrt{2})\uparrow$	r−1	2(r−1)	−1	0
弧	N	r−1	2(r−1)	−1	M
四辺形	3	A*	B*	−1	A*
8×8ドット グラフィックス文字	7	−	−	−	−
8×8ドット以外のグラフィックス文字，塗りつぶし	B**	A**	−	−	−
1文字 1ドット	−	−	−	−	−

△X：X座標変位

△Y：Y座標変位

r　：半径

N　：描画総ドット数

M　：マスキング・ドット数

↑　：切り上げ

A*　：第1描画方向変位数

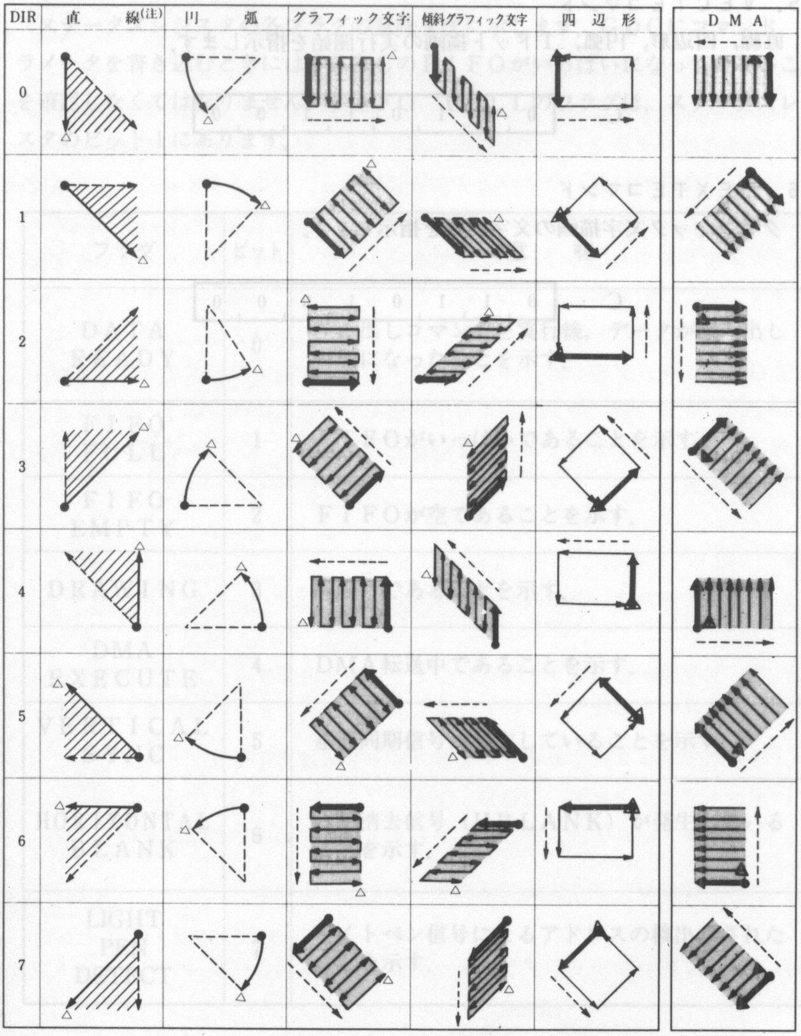
B*　：第2描画方向変位数

A**：第1描画方向ドット数

B**：第2描画方向ドット数−1

−　：設定禁止パラメータ

DIR (Direction)



●：描画始点　　▨：定義域

→：描画方向　　□：描画域

△：描画終了時のEAD指示点

➡：第1方向

⇢：第2方向

(注) 直線のDIR定義中，破線矢印の方向へは書けません。
　　また，描画終了時のEAD指示点は描画した延長上にきます。

5．VECTEコマンド

直線，四辺形，円弧，1ドット描画の実行開始を指示します．

C	0	1	1	0	1	1	0	0

6．TEXTEコマンド

グラフィック文字描画の文字開始を指示します．

C	0	1	1	0	1	0	0	0

□ステータスレジスタ

ステータスレジスタの各フラグの意味を次に示します．ＧＤＣにコマンド，パラメータを書き込むときには，ＧＤＣのＦＩＦＯがいっぱいになっていないことを確認しなくてはなりません．ＦＩＦＯ　ＦＵＬＬのフラグは，ステータスレジスタのビット１にあります．

フラグ	ビット	意　味
DATA READY	0	読み出しコマンドを実行後，データが読み出し可能になったことを示す．
FIFO FULL	1	ＦＩＦＯがいっぱいであることを示す．
FIFO EMPTY	2	ＦＩＦＯが空であることを示す．
DRAWING	3	描画中であることを示す．
DMA EXECUTE	4	DMA転送中であることを示す．
VERTICAL SYNC	5	垂直同期信号が発生していることを示す．
HORIZONTAL BLANK	6	水平消去信号（ＨＢＬＡＮＫ）が発生していることを示す．
LIGHT PEN DETECT	7	ライトペン信号によるアドレスの検出がされたことを示す．

ワンポイント・ヒント

★グラフィック文字の拡大描画

ＧＤＣには，ＺＯＯＭコマンドがあり，拡大表示，拡大描画の機能があります．拡大表示とは，グラフィック画面の一部を大きく拡大して表示することですが，拡大表示には外付けのハードウェアが必要であり，ＰＣ－９８００シリーズは残念ながらサポートしていません．拡大描画とは，始めから図形を拡大してグラフィックメモリに書き込んでいくことであり，ＰＣ－９８００シリーズでも実現できます．ただ，ＢＡＳＩＣ，ＬＩＯ，ＢＩＯＳではこの機能をサポートしていないので，直接ＧＤＣを制御する必要があります．以下はその方法の紹介です．

ＺＯＯＭコマンドのパラメータの下位４ビットが拡大描画係数です．０が１倍，１が２倍，２が３倍，最大は１５で１６倍です．拡大描画係数は一度設定されると再設定されるまで有効で以後行なわれるグラフィック文字の描画コマンドに対して働きます．

手　順

(1) ＴＥＸＴＷコマンドで描くべき８×８の文字パターンをセットする．

(2) Ｘ，Ｙ座標からメモリ上のＧＤＣアドレスを計算する．Ｘ，Ｙをスクリーン座標，Ｐをプレーン番号（１，２，３）とするとアドレスは次のように求まる．

ワードアドレス＝Ｙ＊４０＋Ｘ￥１６＋Ｐ＊＆Ｈ４０００
ドットアドレス＝Ｘ　ＭＯＤ　１６
ドットアドレスとは，ＧＤＣアドレス１ワードの中での位置をいう．

(3) ＣＳＲＷコマンドで描画開始アドレスをセットする．

(4) ＶＥＣＴＷコマンドでグラフィック文字の描画の各パラメータをセットする．

(5) ＺＯＯＭコマンドで拡大描画係数をセットする．

(6) ＴＥＸＴＥコマンドで描画の実行を指示する．

プログラム

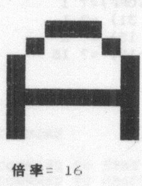
このプログラムはＺＯＯＭコマンドのデモンストレーション・プログラムで，

“A”という文字を倍率を１倍から１６倍まで変えて画面に表示するものです．上に述べたアルゴリズムをＢＡＳＩＣで実行しています．行番号３５０では，ＶＥＣＴＷコマンドのパラメータで描画方向等を出力していますが，下位３ビットが描画方向を意味しており，＆Ｈ１０，＆Ｈ１３，＆Ｈ１７などと変えてみると文字の角度が変わります．

```
100 '拡大描画デモンストレーションプログラム
110 SCREEN 3,1
120 FOR I=0 TO 15
130    OUT &HA2,&H78    :'TEXTW COMMAND
140    OUT &HA0,&H0     :'00000000
150    OUT &HA0,&H38    :'00111000
160    OUT &HA0,&H44    :'01000100
170    OUT &HA0,&H82    :'10000010
180    OUT &HA0,&H82    :'10000010
190    OUT &HA0,&HFE    :'11111110
200    OUT &HA0,&H82    :'10000010
210    OUT &HA0,&H82    :'10000010
220  '
230    X=300:Y=180      :'描画開始座標
240    AD=Y*40+X¥16     :'GDC アドレスへ変換
250    ADL=AD MOD 256   :'下位バイト
260    ADH=AD¥256+&H40  :'上位バイト
270    DAD=X MOD 16     :'ドットアドレス
280  '
290    OUT &HA2,&H49    :'CSRW COMMAND
300    OUT &HA0,ADL
310    OUT &HA0,ADH
320    OUT &HA0,DAD*16
330  '
340    OUT &HA2,&H4C    :'VECTW COMMAND
350    OUT &HA0,&H12    :'DIR=2
360    OUT &HA0,&H7
370    OUT &HA0,&H0
380  '
390    OUT &HA2,&H46    :'ZOOM COMMAND
400    OUT &HA0,I       :'拡大描画係数
410  '
420    CLS 3
430    LOCATE 40,14:PRINT "倍率=";I+1
440  '
450    OUT &HA2,&H68    :'TEXTE COMMAND
460    FOR J=1 TO 1000:NEXT
470 NEXT
```

実行結果

倍率が１６のときの画面のコピーです．



倍率= 16

Ok

ワンポイント・ヒント

★カーソルの形状を変更する方法

ＧＤＣのＣＳＲＦＯＲＭコマンドを用いれば，カーソルのブリンキング速度，大きさなどを変えることができます．（1．3参照）しかし，この機能はＢＩＯＳではサポートされていないので，直接ＧＤＣを制御します．

ここでは，わかりやすいようにＢＡＳＩＣで記述してみました．実際に動かして試してみて下さい．BLINK RATEは値が大きいほどブリンキングはゆっくりになります．

プログラム

```
100 'カーソル形状を変更する
110 INPUT "CURSOR (0:OFF, 1:ON)=";CS
120 INPUT "BLINK  (0:ON, 1:OFF)=";BD
130 INPUT "BLINK RATE (1 - 31) =";BL
140 INPUT "START LINE (0 - 15) =";CST
150 INPUT "FINISH LINE (0 -15) =";CFI
160 OUT &H62,&H4B
170 OUT &H60,CS*&H80
180 OUT &H60,(BL MOD 4)*&H40+BD*&H20+CST
190 OUT &H60,CFI*8+BL¥4
```

入力例

(1) アンダーライン・カーソルです．かなり速くブリンキングします．

```
run
CURSOR (0:OFF, 1:ON)=? 1
BLINK  (0:ON, 1:OFF)=? 0
BLINK RATE (1 - 31) =? 4
START LINE (0 - 15) =? 14
FINISH LINE (0 -15) =? 14
Ok
```

(2) ブロック・カーソルです．ブリンキングはしません．

```
run
CURSOR (0:OFF, 1:ON)=? 1
BLINK  (0:ON, 1:OFF)=? 1
BLINK RATE (1 - 31) =? 1
START LINE (0 - 15) =? 0
FINISH LINE (0 -15) =? 15
Ok
```

ワンポイント・ヒント

★640×409の解像度を実現する方法

1．1で述べたようにグラフィックVRAMは1プレーンあたり32768バイトありますが，640×400の場合に実際に使うのは32000バイトです．768バイト余っていますので，これも使うとすれば理屈ではあと9ライン増やせるはずです．409ラインを表示するにはGDCとSYNCコマンドで表示ライン数（L／F）を409にしてやれば良さそうです．9ラインぐらいの増加ならディスプレイもなんとか同期してくれるでしょう．ただ，気をつけなくてはいけないのは，テキスト用のGDCも同様にライン数をふやしてやることです．

ここでは，わかりやすいようにBASICでプログラムを書いてみました．始めにグラフィック用GDCに対してSYNC，および，PITCHコマンドを与えます（130－270）．次に同様にテキスト用GDCにも与え（300－400），最後に両方のGDCにSTARTコマンドを与えます（410－420）．440行では通常の640×400の範囲を緑のボックスで示します．新しく作った9ラインにはBASICのLINE等では書き込めませんから，POKEで直接書き込んで表示されていることを確認しています（450－480）．

プログラム

```
100 '640X409の解像度を実現する
110 SCREEN 3
120 '
130 'SET GRAPHIC GDC
140 OUT &HA2,&HE       :'SYNC COMMAND
150 OUT &HA0,&H16
160 OUT &HA0,&H26
170 OUT &HA0,&H3
180 OUT &HA0,&H11
190 OUT &HA0,&H3
200 OUT &HA0,&H7
210 OUT &HA0,&H99      :'NORMAL=&h90
220 OUT &HA0,&H65
230 OUT &HA2,&H47      :'PITCH COMMAND
240 OUT &HA0,&H28
250 OUT &HA2,&H72
260 OUT &HA0,&HF0
270 OUT &HA0,&H3F
280 '
290 'SET TEXT GDC
300 OUT &H62,&HE       :'SYNC COMMAND
310 OUT &H60,&H10
320 OUT &H60,&H4E
330 OUT &H60,&H7
340 OUT &H60,&H25
350 OUT &H60,&H7
360 OUT &H60,&H7
370 OUT &H60,&H99      :'NORMAL=&h90
380 OUT &H60,&H65
390 OUT &H62,&H47      :'PITCH COMMAND
400 OUT &H60,&H50
410 OUT &H62,&HD       :'START COMMAND TO TEXT GDC
420 OUT &HA2,&HD       :'START COMMAND TO GRAPHIC GDC
430 '
440 LINE(0,0)-(639,399),4,B:'640X400の範囲
450 DEF SEG=&HA800     :'青のプレーン
460 FOR I=0 TO 719     :'新しく作った9ラインを
470   POKE 32000+I,&HFF :'&HFFで埋める
480 NEXT
```

I．4　モードフリップフロップコントロール

ＣＲＴ関係の各種の設定を行うフリップフロップです．Ｉ／Ｏアドレスの６８Ｈにあり，８種類の設定ができます．対象となるフリップフロップの選択は，ＡＤＲ０～２で行い，ＤＴに値をセットします．

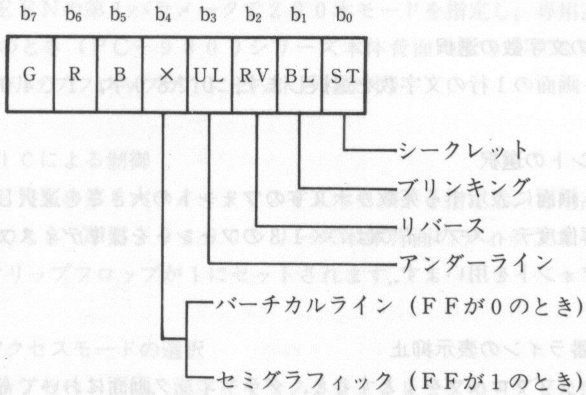
	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
６８Ｈ					AD R２	AD R１	AD R０	DT

ＡＤＲ０～２と対応するフリップフロップは次のようになります．

<table>
<tr><th rowspan="2"></th><th rowspan="2">AAA
DDD
RRR
210</th><th colspan="2">DT</th></tr>
<tr><th>1</th><th>0</th></tr>
<tr><td rowspan="2">0</td><td rowspan="2">000</td><td colspan="2">アトリビュートのビット４の意味</td></tr>
<tr><td>セミグラフィック</td><td>バーチカルライン</td></tr>
<tr><td rowspan="2">1</td><td rowspan="2">001</td><td colspan="2">グラフィックモード</td></tr>
<tr><td>白黒</td><td>カラー</td></tr>
<tr><td rowspan="2">2</td><td rowspan="2">010</td><td colspan="2">１行の文字数</td></tr>
<tr><td>４０字</td><td>８０字</td></tr>
<tr><td rowspan="2">3</td><td rowspan="2">011</td><td colspan="2">文字フォント</td></tr>
<tr><td>７×１３</td><td>６×８</td></tr>
<tr><td rowspan="2">4</td><td rowspan="2">100</td><td colspan="2">奇数番ラインの表示抑止</td></tr>
<tr><td>あり</td><td>なし</td></tr>
<tr><td rowspan="2">5</td><td rowspan="2">101</td><td colspan="2">漢字アクセスモード</td></tr>
<tr><td>ドットマップ</td><td>コードアクセス</td></tr>
<tr><td rowspan="2">6</td><td rowspan="2">110</td><td colspan="2">不揮発性メモリへの書き込み</td></tr>
<tr><td>許可</td><td>禁止</td></tr>
<tr><td rowspan="2">7</td><td rowspan="2">111</td><td colspan="2">画面表示</td></tr>
<tr><td>ON</td><td>OFF</td></tr>
</table>

(０)アトリビュートのビット４の意味

実アドレスのＡ２０００ＨからＡ３ＦＦＦＨにはテキスト画面のアトリビュート（属性）エリアがあり，１バイトが１文字に対応し，属性を表わしています．アトリビュートの各バイトは次のような構成になっており，ビット４の意味は，モードフリップフロップの設定で変化するようになっています．



(1)　カラー／白黒の選択

グラフィック画面に対してカラーか白黒かを選択します．白黒を選択すると，カラーパレットの緑の出力のＯＮ／ＯＦＦによって白または黒（つまり３色ともＯＮ／ＯＦＦする）となります．この切り換えは，パレットレジスタの出力に対して行われるので，グラフィック画面に対してのみ有効であり，テキスト画面には，影響を及ぼしません．

●機械語による直接制御

白黒の指定	ＭＯＶ	ＡＬ，３
	ＯＵＴ	６８Ｈ，ＡＬ
カラーの指定	ＭＯＶ	ＡＬ，２
	ＯＵＴ	６８，ＡＬ

●ＢＩＯＳによる制御

２．３のグラフィックスクリーンのモード設定を使います．レジスタＣＨのビ

ット5が，白黒／カラーの切り換えを意味します．

●ＬＩＯによる制御

ＧＳＣＲＥＥＮの第１パラメータで指定します．

●ＢＡＳＩＣによる制御

ＳＣＲＥＥＮ文の第１パラメータで指定します．

(2)　１行の文字数の選択

テキスト画面の１行の文字数を選択します．０で８０字，１で４０字です．

(3)　フォントの選択

テキスト画面に表示する英数カナ文字のフォントの大きさを選択します．通常は専用高解像度ディスプレイでは７×１３のフォントを標準ディスプレイでは，６×８のフォントを用います．

(4)　奇数番ラインの表示抑止

このフリップフロップを１とすると，グラフィック画面において奇数番目のラインは表示されずブランクとなります．この機能は専用高解像度ディスプレイを用いて２００本表示するときのみに用いられます．専用高解像度ディスプレイでは，４００ラインの表示ができますが，ＧＶＲＡＭの２００行分をちょうど画面いっぱいに表示するために，ＧＤＣの設定により１行を２ラインとしています．ただし，その場合は横線が太くなってしまうので，それを自然に見えるようにするため，１ラインおきにブランクとなるようにしているのです．

●機械語による直接制御

奇数番ラインの表示制御ありの指定

```
MOV  AL, 9
OUT  68H, AL
```

奇数番ラインの表示制御なしの指定

```
MOV  AL, 8
OUT  68H, AL
```

●ＢＩＯＳによる制御

２．３のグラフィックスクリーンのモード設定が関係します．レジスタＣＨのビット７，６が，０１または１０（つまり２００本モード）のときで，専用高解像度ディスプレイのとき（ＰＣ－９８００シリーズ本体背面のディップスイッチで指定），このフリップフロップが１にセットされます．

●ＬＩＯによる制御

ＧＳＣＲＥＥＮの第１パラメータで２００本モードを指定し，専用高解像度ディスプレイのとき（ＰＣ－９８００シリーズ本体背面のディップスイッチで指定），このフリップフロップが１にセットされます．

●ＢＡＳＩＣによる制御

ＳＣＲＥＥＮ文の第１パラメータで２００本モードを指定し，専用高解像度ディスプレイのとき（ＰＣ－９８００シリーズ本体背面のディップスイッチで指定），このフリップフロップが１にセットされます．

(5)　漢字アクセスモードの選択

ドットマップモードでは，漢字キャラクタジェネレータのアクセスはいつでもできますが，コードアクセスモードでは，ＶＳＹＮＣ（垂直同期信号期間）中に行なわなくてはなりません．

(6)　不揮発性メモリへの書き込みの許可，禁止

ＢＡＳＩＣのＭＯＮコマンドのＳＳＷで読み書きするメモリスイッチは，実アドレスＡ３ＦＥ２Ｈ～Ａ３ＦＦ６Ｈにありますが，このメモリは通常書き込み禁止状態にあり，このメモリへ書き込むためには，このフリップフロップを１にして書き込み許可の状態にしなければなりません．

(7)　画面表示のＯＮ／ＯＦＦ

このフリップフロップを０にすると画面は全く表示されなくなります．これは電源投入時に不確定状態のビデオＲＡＭの内容が表示されるのを抑制するのに使われます．パワーオンによるリセット信号でこのフリップフロップの初期状態は０であり画面表示はＯＦＦとなります．ＣＲＴ関係のハードウェアの初期設定の後，このフリップフロップは１とされます．

ワンポイント・ヒント

★メモリスイッチを書き換える方法

ＰＣ－９８００シリーズには，メモリスイッチと呼ばれる不揮発性のＲＡＭがあって，通信のパラメータ等を設定するようになっています．普通はMONコマンドでモニタに入ってＳＳＷ１などと入力して書き換えます．しかし，端末プログラムなどの作成にあたっては，ＢＡＳＩＣから通信速度（ボーレート）などを設定できれば便利でしょう．

メモリスイッチは主記憶空間の実アドレスＡ３ＦＥ２Ｈ～Ａ３ＦＦ６Ｈにあって，各スイッチとの対応は次の表のとおりです．しかし，このアドレスから，ＰＥＥＫで値を読むことはできますが，ＰＯＫＥしても内容は書き換わりません．それは，このＲＡＭが書き込み禁止状態となっているからです．書き込み禁止状態を解除するにはモードフリップフロップを制御します（１．４参照）．Ｉ/Ｏアドレス６８Ｈに０ＤＨを書き込むと解除になります．変更が終わったら，０ＣＨを書き込んでまた禁止にしておきます．

スイッチ	アドレス
ＳＷ１	Ａ３ＦＥ２
ＳＷ２	Ａ３ＦＥ６
ＳＷ３	Ａ３ＦＥＡ
ＳＷ４	Ａ３ＦＥＥ
ＳＷ５	Ａ３ＦＦ２
ＳＷ６	Ａ３ＦＦ６

プログラム

このプログラムはＢＡＳＩＣから，メモリスイッチの内容を書き変えるものです．まず，スイッチ番号を入力します．現在の内容を表示しますから，次に書き込む値を入力して下さい．

```
10 DEF SEG=&HA000
20 INPUT "ｽｲｯﾁ ﾊﾞﾝｺﾞｳ=":SW
30 ADRS=&H3FDE+SW*4
40 PRINT "ｹﾞﾝｻﾞｲﾉ ｱﾀｲ=&H";HEX$(PEEK(ADRS))
50 INPUT "ｱﾀﾗｼｲ     ｱﾀｲ=":K
60 OUT &H68,&HD:'書き込み禁止を解除
70 POKE ADRS,K :'ﾒﾓﾘｽｲｯﾁ ﾆ ｶｷｺﾑ
80 OUT &H68,&HC:'書き込み禁止に戻す
```

実行例

```
run
ｽｲｯﾁ ﾊﾞﾝｺﾞｳ=? 1
ｹﾞﾝｻﾞｲﾉ ｱﾀｲ=&H48
ｱﾀﾗｼｲ     ｱﾀｲ=? &h23
Ok
run
ｽｲｯﾁ ﾊﾞﾝｺﾞｳ=? 1
ｹﾞﾝｻﾞｲﾉ ｱﾀｲ=&H23
ｱﾀﾗｼｲ     ｱﾀｲ=? &h48
Ok
```

1．5　ボーダカラーレジスタ

ボーダカラーとは，画面でテキストやグラフィックを表示できる範囲より外の部分につける色のことをいいます．ボーダカラーレジスタと呼ばれる3ビットの書き込み専用のレジスタがあり，画面の縁の部分の走査時には，このレジスタの出力がマルチプレクサで選択されビデオ信号として出力されます．ただし，このボーダカラーの機能は標準解像度のディスプレイに対してのみ正しく働きます．専用高解像度のディスプレイに対して黒以外のボーダカラーを設定すると画面が乱れますので注意して下さい．

ボーダカラーレジスタはI／Oアドレスの6CHにあります．ビット4，5，6がそれぞれ青，赤，緑に対応しています．

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
6CH	0	G	R	B	0	0	0	0

6CHの内容	ボーダカラー
00H	黒
10H	青
20H	赤
30H	紫
40H	緑
50H	水色
60H	黄
70H	白

●機械語による直接制御

例　ボーダカラーに赤を設定します．

```
MOV    AL, 20H
OUT    6CH, AL
```

●ＢＩＯＳによる制御

２．５のボーダカラーの設定を使います．ＢＩＯＳでは専用高解像度のディスプレイかどうかの判定を行い，専用高解像度のときにはボーダカラーレジスタへの設定をしないようになっています．

●ＬＩＯによる制御

３．４のＧＯＬＯＲ１を使います．第１パラメータでボーダカラーを設定します．ＢＩＯＳと同じく専用高解像度のときには，ボーダカラーの設定は行なわれません．

●ＢＡＳＩＣによる制御

ＣＯＬＯＲ文を使います．第３パラメータでボーダカラーを設定します．ＢＩＯＳ，ＬＩＯと同じく専用高解像度のときには，ボーダカラーの設定は行なわれません．

1．6　バンク切り換えスイッチ

ＰＣ－９８０１Ｅ／Ｆ／Ｍでは，グラフィックＶＲＡＭがＰＣ－９８０１の２倍の１９２Ｋバイト実装されています．増えた９６Ｋバイトは元の９６Ｋバイトと同じアドレス上にあります．もとの９６Ｋバイトをバンク０，増えた９６Ｋバイトをバンク１と呼びます．バンク０とバンク１は切り換えてどちらか一方を使うようになっています．この切り換えをバンク切り換えと呼びます．この切り換えによってバンクの選択をするわけですが，表示時と描画時とでアクセスされるバンクを独立に設定できるようになっています．ですから，バンク０を表示しながら，バンク１に描画を行なう，または，その逆も可能となります．

バンク切り換えスイッチは，Ｉ／Ｏアドレスの０Ａ４Ｈと０Ａ６Ｈにあります．０Ａ４Ｈが表示バンクの切り換え，０Ａ６Ｈが描画バンクの切り換えです．

表示バンクの切り換え

０Ａ４Ｈ	０：バンク０
	１：バンク１

描画バンクの切り換え

０Ａ６Ｈ	０：バンク０
	１：バンク１

表示バンクとは，表示のためにアクセスされるときに使われるバンクのことで，描画バンクとは，表示以外の目的でアクセスされるときに使われるバンクのことです．ＣＰＵ，ＧＤＣからの，読み込み，書き込みはこれに相当します．

ワンポイント・ヒント

★スムーズ・スクロールの方法

スムーズ・スクロールとは，テキスト画面の文字をスクロールするときに，１文字単位の大きさで上下するのではなく，１ドット単位で上下させることをいいます．画面上の文字は流れるように上下するので，文字を目で追うことができ，非常に見やすくなります．ＰＣ－９８０１のN88－ＢＡＳＩＣ，ＢＩＯＳ等では，この機能をサポートしていませんが，ハード的には実現可能です．そこで，このスムーズスクロールを実現する方法を紹介します．

ＣＲＴの制御用ＬＳＩ，ＣＲＴ　Ｍ／Ｓ（マスタスライス）　（μＰＤ５２６１１）にはスムーズスクロール用にＳＳＬ，ＳＵＲ，ＳＤＲの３つのレジスタがあり，画面上の任意の場所に１つだけスムーズスクロール·エリアを設定できます．次に各レジスタのＩ／Ｏアドレスと意味を示します．

レジスタ	Ｉ／Ｏアドレス	意　味
ＳＳＬ	７６Ｈ	スムーズスクロール・エリア内のスクロールライン数を設定する．
ＳＵＲ	７８Ｈ	０－（スムーズスクロールエリアよりの上の部分の行数）を設定する．
ＳＤＲ	７ＡＨ	（スムーズスクロールエリアの行数－１を設定する．

ＳＵＲとＳＤＲでスクロールエリアを指定し，ＳＳＬでスクロールライン数を指定します．スクロールエリア内の文字はＳＳＬで指定したライン数だけ上へずれて表示されます．スムーズスクロールは，ＳＳＬを順次増減させることによって実現されます．ただし，ＳＳＬによってずらすことのできるライン数は１行のライン数までですから，連続してスムーズスクロールを行なうには，通常のテキスト画面のスクロールと組み合わせて行なう必要があります．

プログラム

次のＢＡＳＩＣプログラムは，連続スムーズスクロールのサンプル・プログラムです．このプログラムでは，画面分割をして２５行目を裏画面（０Ａ２０００Ｈ～）としていますが，これは，行番号２４０のＰＲＩＮＴで画面に表示される文字を見えなくするためです．スクロールエリアは画面全体です．行番号２６０でＳＳＬを順次インクリメントして，１ラインごとのスクロールをしています．２５行モードでは，１行は１６ラインですので，ＳＳＬを１６までインクリメントしたら，つぎは，ＳＳＬを０に戻し，同時にVRAM上の文字の方を１行スクロールします(行番号２８０)．画面がちらつかないようにこれらはVSYNCの期間中に行ないます．

```
10 '***************************************************
20 '*       SMOOTH SCROLL DEMONSTRATION PROGRAM        *
30 '***************************************************
100 CONSOLE 0,25,0
110 LOCATE 0,25
120 OUT &H78,0           :'ｽｸﾛｰﾙ ﾉ ｾﾝﾄｳｦ ﾀﾞｲ 1ｷﾞｮｳ ｶﾗ
130 OUT &H7A,24          :'ｽｸﾛｰﾙ ｷﾞｮｳｽｳ ｦ 25 ﾄ ｽﾙ
135 '
140 OUT &H62,&H70        :'GDC ﾉ ｶﾞﾒﾝ ﾌﾞﾝｶﾂ
150 OUT &H60,0           :'    ﾀﾞｲ 1 ｶﾞﾒﾝ  ﾊ
160 OUT &H60,&H0         :'        1--24 ｷﾞｮｳ
170 OUT &H60,&H0
180 OUT &H60,&H18
185 '
190 OUT &H60,&H0         :'    ﾀﾞｲ 2 ｶﾞﾒﾝ  ﾊ
200 OUT &H60,&H8         :'         25 ｷﾞｮｳ
210 OUT &H60,&H0
220 OUT &H60,1
225 '
230 FOR I=1 TO 1000
240    PRINT I,I*I;
250    FOR J=1 TO 15
255       '   VSYNC ﾉ ﾄｷﾆ ｽｸﾛｰﾙ ﾗｲﾝ ｽｳ ｦ ﾍﾝｺｳ
260       IF INP(&H60)AND &H20 THEN OUT &H76,J ELSE 260
270    NEXT
275    '  VSYNC ﾉ ﾄｷﾆ ｽｸﾛｰﾙ ﾗｲﾝ ｽｳ ｦ 0 ﾆｼ､ 1 ｷﾞｮｳ ｽｸﾛｰﾙ
280    IF INP(&H60)AND &H20 THEN OUT &H76,0:PRINT ELSE 280
290 NEXT
```

CRTの制御用LSI、CRT M/S（マスタスライス1）にはスムーズスクロール用にSSL、SUR、SDRの3つのレジスタがあり、画面上の任意の[illegible]できます。次に各レジスタの I/Oアドレスと意味を示します。

レジスタ	I/Oアドレス	意味
SSL	76H	スムーズスク[illegible]のスクロール[illegible]ライン[illegible]
SUR	78H	0～（スム[illegible]ム）～0 [illegible]する。
SDR	7AH	[illegible]行[illegible]－1を設定する。

SURとSDRでスクロールエリアを指定し、SSLでスクロールライン数を指定します。スクロールエリア内の文字はSSLで指定したライン数だけ上へずれて表示されます。スムーズスクロールは、SSLを順次増減させることによって実現されます。ただし、SSLによってずらすことのできるライン数は1行のライン数までですから、連続してスムーズスクロールを行なうには、通常のテキスト画面のスクロールと組み合わせて行なう必要があります。

第2章

グラフィックBIOS

２．０　ＢＩＯＳの概要

ＰＣ－９８００シリーズでは，ハードウェアを扱う基本的な入出力ルーチンがＢＩＯＳとしてまとめられています．ここでは，その中でグラフィック関係のものを取り上げます．グラフィック関係のＢＩＯＳは次のように１１種類あります．

ＢＩＯＳルーチン一覧表

	ＡＨの値	ＢＩＯＳルーチン	制御するハードウェア
1	４０Ｈ	グラフィック画面の表示開始	ＧＤＣ
2	４１Ｈ	グラフィック画面の表示停止	ＧＤＣ
3	４２Ｈ	グラフィック・スクリーンのモード設定	モードフリップフロップ
4	４３Ｈ	カラー・パレットの設定	パレットレジスタ
5	４４Ｈ	ボーダ・カラーの設定	ボーダーカラーレジスタ
6	４５Ｈ	グラフィック・スクリーンへのドット書き込み	ＧＶＲＡＭ
7	４６Ｈ	グラフィック・スクリーンからのドット読み出し	ＧＶＲＡＭ
8	４７Ｈ	直線および四角形の描画	ＧＤＣ
9	４８Ｈ	円弧の描画	ＧＤＣ
10	４９Ｈ	グラフィクス文字の描画	ＧＤＣ
11	４ＡＨ	高速書き込みモードの選択	ＧＤＣ

描画関係のＢＩＯＳ（６－１０）のうち，６と７は直接グラフィックRAMにアクセスするのに対して，他はGDCを使って描画しています．

ＢＩＯＳは描画においてスクリーン座標が使えるものの，アクティブ画面，ディスプレイ画面などの考え方がＬＩＯ，ＢＡＳＩＣなどとは異なり，かなりハードウェアに近いものとなっています．ＢＩＯＳを用いた描画を行なう際には，その点をよく理解していただくことが必要です．

□ＢＩＯＳにおける画面の考え方

●ディスプレイ画面

ＰＣ－９８００シリーズのもつグラフィックＶＲAMは実アドレス，Ａ８０００－ＡＦＦＦＦ，Ｂ００００－Ｂ７ＦＦＦ，Ｂ８０００－ＢＦＦＦＦにそれぞれ１プレーンが配置され，各プレーンの出力はパレットレジスタによる変換を受けた後，ＢＲＧの各出力となります．

ＢＩＯＳにおいてはグラフィックスクリーンのモード設定で表示範囲の設定ができます．表示範囲には，ＡＬＬ，ＬＯＷＥＲ，ＵＰＰＥＲの３種類があり，ＡＬＬとは，６４０×４００の解像度を意味し，各プレーンの全領域を表示範囲とすることです．ＬＯＷＥＲまたはＵＰＰＥＲは，各プレーンの半分の領域を表示範囲とすることで，ＬＯＷＥＲは下半分を，ＵＰＰＥＲは上半分を意味します．

表示に使われる領域

表示範囲(解像度)	プレーン1	プレーン2	プレーン3
ＡＬＬ (640×400)	A8000-AFCFF	B0000-B7CFF	B8000-BFCFF
ＬＯＷＥＲ(640×200)	A8000-ABE7F	B0000-B3E7F	B8000-BBE7F
ＵＰＰＥＲ(640×200)	ABE80-AFCFF	B3E80-B7CFF	BBE80-BFCFF

各プレーンは常にパレットレジスタによって合成されて出力されます．白黒モードにおける単一プレーンのみの出力は，パレットレジスタをうまく設定することによって始めて実現されます．

●アクティブ画面

描画の際の領域の指定は，描画画面ナンバ，描画画面範囲および解像度モードで行ないます．描画画面ナンバによってプレーンを指定し，描画画面範囲と解像度モードの組み合わせによってＡＬＬ，ＬＯＷＥＲ，ＵＰＰＥＲを指定します．描画画面範囲と解像度モードの両方を１とすることは許されません．

	描画画面範囲	解像度モード
ＡＬＬ	0	1
ＬＯＷＥＲ	0	0
ＵＰＰＥＲ	1	0

カラーで使う場合は描画画面ナンバを１１とし，[ＢＸ＋０]にパレット番号を設定するのが一般的です．

●ＢＩＯＳの呼び出し方法

ＡＨに値をセットして，システムコール　ＩＮＴ　１８Ｈで呼び出します．ＢＩＯＳではレジスタはすべて保存されます．引数エリアは，ＤＳ：ＢＸで指示すればよいので，任意の領域に置くことができます．ＢＩＯＳルーチンによっては，引数エリアの後方十数バイトを作業領域として使うものがあります．ＢＡＳＩＣ，ＬＩＯでは引数エリアとしてセグメント６０Ｈの６４０Ｈからを使っています．

●メモリマップ

BIOSは，セグメントF990Hの2C00Hからにあります．

アドレス	内容
F990H：0000H	グラフィックLIO
2C00H	各ルーチンへの分岐部
2C43H	グラフィック画面の表示開始
2C5FH	グラフィック画面の表示停止
2C79H	グラフィックスクリーンのモード設定
2D00H	パレットレジスタの設定
2D33H	ボーダ・カラーの設定
2D4AH	グラフィックスクリーンへのドット列の書込み
2EEAH	グラフィックスクリーンのドット列の読み出し
2FB7H	直線および四角形，円弧の描画
32C3H	グラフィック文字の描画
3467H	高速書き込みモードの選択

２．１　グラフィック画面の表示開始

機　能　グラフィック画面の表示を開始します．

割　込　ＩＮＴ　１８Ｈ

絶対番地　Ｆ９９０Ｈ：２Ｃ４３Ｈ

入力条件　ＡＨ←４０Ｈ

出力条件　なし

破　壊　なし

解　説

ＧＤＣに対してグラフィック画面の表示開始コマンド（０ＤＨ）を与え，システム共通領域ＰＲＸＣＲＴ（００００Ｈ：０５４ＣＨ）のビット７をセットします．このビットは，グラフィック画面が表示中であることを示すものです．

処　理

(1) ＧＤＣのステータスフラグをセンスし，垂直同期信号が発生し，かつ，ＧＤＣのＦＩＦＯがfullでなくなるまで待つ．

```
WAIT:IN    AL, 0A0H
     XOR   AL, 20H
     TEST  AL, 22H
     JNE   WAIT
```

(2) ＧＤＣのコマンドレジスタに画面の表示開始コマンド（０ＤＨ）を与える．

```
     MOV   AL, 0DH
     OUT   0A2H, AL
```

(3) ＢＩＯＳのワークエリアＰＲＸＣＲＴ（００００Ｈ：０５４ＣＨ）のビット７をセットする．

2．2　グラフィック画面の表示停止

機　能	グラフィック画面の表示を停止します．
割　込	ＩＮＴ　１８Ｈ
絶対番地	Ｆ９９０Ｈ：２Ｃ５ＦＨ
入力条件	ＡＨ←４１Ｈ
出力条件	なし
破　壊	なし

解　説

ＧＤＣに対してグラフィック画面の表示停止コマンド（０ＣＨ）を与え，システム共通領域ＰＲＸＣＲＴ（０００００Ｈ：０５４ＣＨ）のビット7をリセットし，グラフィック画面が表示中でないことを示します．

処　理

(1) ＧＤＣのステータスフラグをセンスし，ＧＤＣのＦＩＦＯがfullでなくなるまで待つ．

```
WAIT：IN    AL, 0A0H
      TEST  AL, 02H
      JNE   WAIT
```

(2) ＧＤＣのコマンドレジスタに画面の表示停止コマンド（０ＣＨ）を与える．

```
      MOV   AL, 0CH
      OUT   0A2H, AL
```

(3) ＢＩＯＳのワークエリアＰＲＸＣＲＴ（００００Ｈ：０５４ＣＨ）のビット7をリセットする．

2．3　グラフィックスクリーンのモード設定

機　能	グラフィック画面のカラー／白黒，表示領域，および分解能などの設定を行ないます．
割　込	INT 18H
絶対番地	F990H：2C79H
入力条件	AH←42H CH←画面モード等

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
CH					0	0	0	0

b_4：表示バンクの指定（E／F／Mのみ）

0：バンク0を使用

1：バンク1を使用

b_5：カラー／白黒の指定

0：カラー

1：白黒

b_7 b_6：VRAM領域の指定

0　1　UPPER

1　0　LOWER

1　1　ALL

出力条件	なし
破　壊	なし

解　説

表示バンクの指定，カラー／白黒の指定，VRAM領域の指定を行ないます．PC−9801E／F／Mの場合は，同じアドレス上に2組のグラフィックVRAMを持ち，バンク切り換えで，どちらか一方の表示または描画が可能です．BIOSでは，表示バンクの設定のみを行なうことができます．描画バンクの切り換えは，直接I／Oポートを制御する（1．5参照）かLIOを使う（3．2参照）必要があります．PC−9801では，表示バンクの指定はできません．0を入力して下さい．

ＶＲＡＭ領域の指定には，ＬＯＷＥＲ，ＵＰＰＥＲ，ＡＬＬの３種類があります．ＬＯＷＥＲおよびＵＰＰＥＲは，ＶＲＡＭ上の３２ＫＢを下半分，上半分の１６ＫＢずつに分け，その一方を使って６４０×２００の分解能を実現します．ＡＬＬは，３２ＫＢ全体を使って６４０×４００の分解能を実現します．

処　理

(1) ＧＤＣのステータスフラグをセンスし，垂直同期信号が発生し，かつ，ＧＤＣのＦＩＦＯが空になるまで待つ．

```
WAIT：IN    AL，0A0H
      XOR   AL，24H
      TEST  AL，24H
      JNE   WAIT
```

(2) Ｉ／Ｏポート（Ａ４Ｈ）にＣＨのビット４に従って表示バンクをセットする．

```
      OUT   0A4H，AL（0または1）
```

(3) モードフリップフロップレジスタ（６８Ｈ）にＣＨのビット５に従って，カラー／白黒をセットする．

```
      OUT   68H，AL
```

ただし，ＡＬの値は　カラー………０２Ｈ
　　　　　　　　　　白黒…………０３Ｈ

(4) モードフリップフロップレジスタ（６８Ｈ）にディスプレイの解像度および，分解能から決まる値をセットする．ディスプレイの解像度は，システム共通領域のＰＲＸＣＲＴ（００００Ｈ：０５４ＣＨ）のビット６にある．

```
      OUT   68H，AL
```

ただし，ＡＬの値は　高解像度（４００）……０８Ｈ
　　　　　　　　　　高解像度（２００）……０９Ｈ
　　　　　　　　　　標　準（２００）……０８Ｈ

(5) 高解像度ディスプレイの場合，分解能に従って，ＧＤＣに１行中の表示ライン数をＣＳＲＦＯＲＭコマンドでセットする．

分解能４００の場合……１ライン／行
分解能２００の場合……２ライン／行

```
      MOV   AL，4BH
```

```
OUT   0A2H, AL
MOV   AL, 0 (or 1) …=ライン数－1
OUT   0A0H, AL
```

(6) GDCに表示開始アドレスおよび表示領域の大きさをSCROLLコマンドでセットする．

〈表示開始アドレス〉

LOWER………………0

UPPER……1F40H (GDCアドレス)

ALL……………………0

GDCアドレスの1は，2バイトに相当するから，1F40Hは，16Kバイトを意味している．

〈表示領域の大きさ〉

高解像度の場合……400ライン

標準の場合…………200ライン

サンプル

(1) 画面モードを640×400のカラーに設定する．

```
MOV  AH, 42H
MOV  CH, 0C0H
INT  18H
```

(2) 画面モードを640×200の白黒に設定する．VRAM領域はLOWERとする．

```
MOV AH, 42H
MOV CH, 0A0H
INT 18H
```

2．4　パレットレジスタの設定

機　能	パレットレジスタにカラーコードを設定する．白黒の場合は，表示画面の合成，選択を指定する．
割　込	ＩＮＴ　１８Ｈ
絶対番地	Ｆ９９０Ｈ：２Ｄ００Ｈ
入力条件	ＡＨ←４３Ｈ ＤＳ←引数ワークエリアのセグメントベース ＢＸ←引数ワークエリアのオフセットアドレス

［ＢＸ＋４～７］にパレットＮＯ．７～０に対するカラーコードを設定する．対応は以下の通りです．

	上位４ビット	下位４ビット
［ＢＸ＋４］	＃６	＃７
［ＢＸ＋５］	＃４	＃５
［ＢＸ＋６］	＃２	＃３
［ＢＸ＋７］	＃０	＃１

出力条件	なし
破　壊	なし
解　説	

［ＢＸ＋４～７］に用意されたカラーコードをパレットレジスタに転送します．カラーモードではカラーコード０～７が次のように各色に対応しています．

０：黒，１：青，２：赤，３：紫，４：緑，５：水色，６：黄，７：白

白黒モードでは，カラーコードは，０または７のみが許され，０のときは，表示ＯＦＦ，１のときは，表示ＯＮを意味します．白黒モードで画面合成を行なう場合は，パレットレジスタは，以下のように設定します．

表示プレーン 1 2 3	パレットレジスタの値 #0	#1	#2	#3	#4	#5	#6	#7
× × ×	0	0	0	0	0	0	0	0
× × ○	0	7	0	7	0	7	0	7
× ○ ×	0	0	7	7	0	0	7	7
○ × ×	0	0	0	0	7	7	7	7
× ○ ○	0	7	7	7	0	7	7	7
○ × ○	0	7	0	7	7	7	7	7
○ ○ ×	0	0	7	7	7	7	7	7
○ ○ ○	0	7	7	7	7	7	7	7

画面合成に関係するプレーンを○印，関係しないプレーンを×印で示します．

処　理

［BX＋4～7］の内容をパレットレジスタ（A8H～AEH）に転送する．ただし，パレットレジスタは，下図のように本ルーチンの引数とは，パレット№の並びが異なるので，順序を入れ替えながら転送する．

レジスタ	上位4ビット	下位4ビット
A8H	#3	#7
AAH	#1	#5
ACH	#2	#6
AEH	#0	#4

サンプル

パレットレジスタを初期状態（パレット番号＝カラーコード）にセットする．

```
MOV  AX, 0060H
MOV  DS, AX
MOV  BX, 640H
MOV  BYTE  [BX+4], 67H
MOV  BYTE  [BX+5], 45H
MOV  BYTE  [BX+6], 23H
MOV  BYTE  [BX+7], 01H
MOV  AH, 43H
INT  18H
```

2．5　ボーダ・カラーの設定

機　能　　ボーダカラーレジスタにボーダカラーを設定する．

割　込　　ＩＮＴ　１８Ｈ

絶対番地　Ｆ９９０Ｈ：２Ｄ３３Ｈ

入力条件　ＡＨ←４４Ｈ

ＤＳ←引数ワークエリアのセグメントベース

ＢＸ←引数ワークエリアのオフセットアドレス

［ＢＸ＋１］←ボーダカラーコード

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
0	G	R	B	0	0	0	0

出力条件　なし

破　壊　　なし

解　説

標準ディスプレイを使用している場合に，ボーダカラーをボーダカラーレジスタ（６ＣＨ）に設定します．専用高解像度ディスプレイを使用している場合には，

何もせず終了します．ボーダカラーとは，画面でテキストやグラフィックを表示できる範囲より外の部分につける色のことをいいます．ボーダカラーコードは，上図のように下位4ビットは，常に0なので，００H，１０H，……，７０Hが，黒，青，……，白に対応します．

処　理

(1) ディスプレイが高解像度かを調べ，つまり，システム共通領域のPRXCRT（００００H：０５４CH）のビット6が1なら，何もせずに戻る．

(2) ボーダカラーレジスタにボーダカラーコードを書き込む．

```
MOV    AL，[BX＋1]
OUT    6CH，AL
```

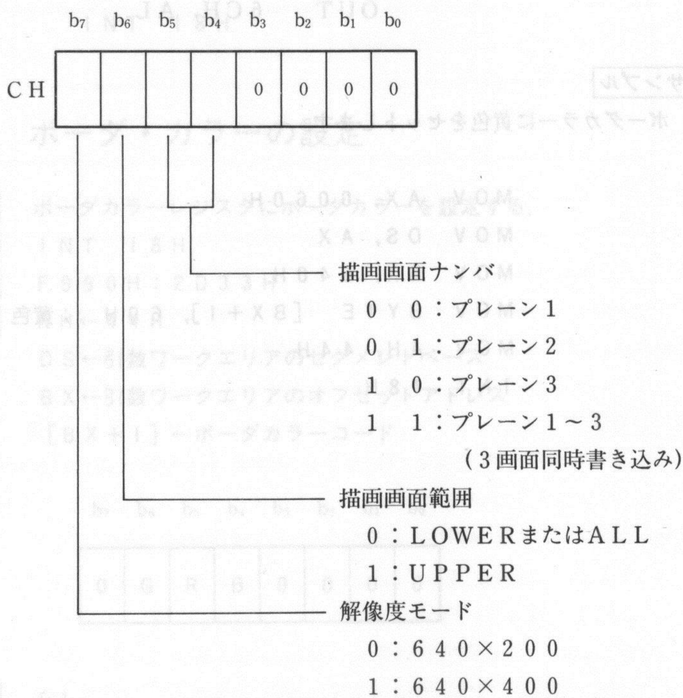
サンプル

ボーダカラーに黄色をセットします．

```
MOV AX，0060H
MOV DS，AX
MOV BX，640H
MOV BYTE  [BX+1]，60H ；黄色
MOV AH，44H
INT 18H
```

2．6　グラフィックスクリーンへの ドット列の書き込み

機　能	グラフィックスクリーンへドット列を書き込みます．
割　込	ＩＮＴ　１８Ｈ
絶対番地	Ｆ９９０Ｈ：２Ｄ４ＡＨ
入力条件	ＡＨ←４５Ｈ ＤＳ←引数ワークエリアのセグメントベース ＢＸ←引画ワークエリアのオフセットアドレス ＣＨ←対象とする描画画面の指定



ＥＳ←ドット・パターン・バッファのセグメントベース

［ＢＸ＋０］←３画面同時書き込みの場合の描画モード

（カラーの場合はパレット番号に相当）

	b7	b6	b5	b4	b3	b2	b1	b0
[ＢＸ＋０]	0	0	0	0	0			

b0 ─ プレーン１
b1 ─ プレーン２
b2 ─ プレーン３

各プレーンに対して
クリアまたはセットを
選択できる．

0：ＣＬＥＡＲ
1：ＳＥＴ

[ＢＸ＋２] ←単一画面書き込みの場合の描画モード

	b7	b6	b5	b4	b3	b2	b1	b0
[ＢＸ＋２]	0	0	0	0	0	0		

下位２ビットで４種類の
モードを選択できる．

0　0：ＲＥＰＬＡＣＥ
0　1：ＣＯＭＰＬＥＭＥＮＴ
1　0：ＣＬＥＡＲ
1　1：ＳＥＴ

[ＢＸ＋８～９] ←ドット列の書き込みを始めるスクリーンＸ座標
（０～６３９の範囲にあること）

[ＢＸ＋０ＡＨ～０ＢＨ] ←ドット列の書き込みを始めるスクリーン
Ｙ座標　（０～１９９or３９９）

[ＢＸ＋０ＣＨ～０ＤＨ] ←書き込むドット列の長さ（ドット数）．

[ＢＸ＋０ＥＨ～０ＦＨ] ←書き込むドットパターンの先頭オフセットアドレス

出力条件　なし

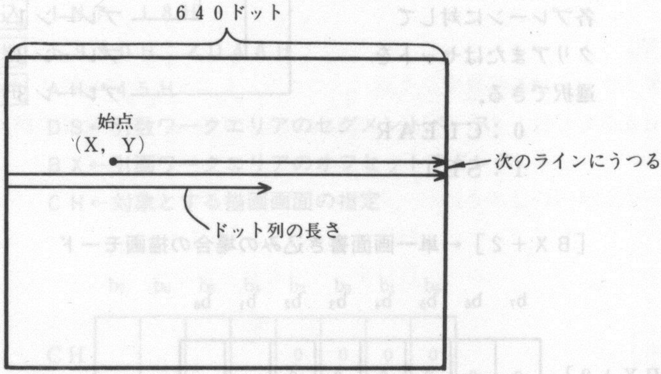
破　　壊　なし

作業領域　[ＢＸ＋２ＡＨ～３９Ｈ]を作業領域として使用しますので，確保して下さい．

解　　説

ＥＳ：[ＢＸ＋０ＥＨ～０ＦＨ] で指されるアドレスから格納されているドッ

トパターンを，［ＢＸ＋８～９］と［ＢＸ＋０ＡＨ～０ＢＨ］で示されるＸ，Ｙ座標を始点として下図のように書き込みます．書き込むドット列の長さは［ＢＸ＋０ＣＨ～０ＤＨ］に与えます．



レジスタＣＨによって対象とする描画画面を指定します．３画面同時書き込み（ＣＨのb_5，b_4が１，１のとき）か否かで選択できる描画モードが異なります．３画面同時の場合はＣＬＥＡＲとＳＥＴのみ，単一画面の場合はＲＥＰＬＡＣＥ，ＣＯＭＰＬＥＭＥＮＴ，ＣＬＥＡＲ，ＳＥＴの４種類を選択できます．以下にそれらの動作を説明します．

(1) ＲＥＰＬＡＣＥ：ドットパターンをそのまま画面に書き込む．

ドットパターン	０ ０ １ １ ０ １ ０ ０ １……
描画結果	０ ０ １ １ ０ １ ０ ０ １……

(2) ＣＯＭＰＬＥＭＥＮＴ：ドットパターンと描画前の状態と排他的論理和（ＸＯＲ）をとって画面に書き込む．

描画前の状態	１ １ １ ０ ０ １ ０ １ ０……
ドットパターン	０ ０ １ １ ０ １ ０ ０ １……
描画結果	１ １ ０ １ ０ ０ ０ １ １……

(3) ＣＬＥＡＲ：ドットパターンが１のところが０となる．ドットパターンが０のところは，描画前の状態×を引き継ぐ．

ドットパターン	０ ０ １ １ ０ １ ０ ０ １……

描画結果　　　　　××００×０××０……

(4) ＳＥＴ：ドットパターンが１のところが１となる．ドットパターンが０のところは，描画前の状態×を引き継ぐ．

ドットパターン　　００１１０１００１……
描画結果　　　　　××１１×１××１……

これらの描画モードは，３画面同時書き込みの場合は［ＢＸ＋０］で，単一画面の場合は［ＢＸ＋２］で指定します．

処　理

本サブルーチンでは，ＧＤＣを使用せず，ＣＰＵから直接グラフィックＶＲＡＭ（０Ａ８０００Ｈ～０ＢＦＦＦＦＨ）を操作することによって，ドット列の書き込みを行っています．

(1) ＧＤＣがＶＲＡＭに書き込み中のときはそれが終了するまで待つ．

```
WAIT1：IN     AL，0A0H      FIFOが空になる
       TEST   AL，04        まで待つ．
       JE     WAIT1
WAIT2：IN     AL，0A0H      描画中フラグが降り
       TEST   AL，08        るまで待つ．
       JNE    WAIT2
```

(2) Ｘ，Ｙ座標からＶＲＡＭ上の描画開始アドレスを計算する．

ＡＤＲＳ＝Ｘ／８＋Ｙ＊８０

(3) 書き込みの開始点および終了点が，ちょうどバイト境界であるとは限らないので，始めと終わりは特別に処理する．その中間では，バイト単位で処理する．

(4) １ビットだけの書き込みは，高速にするため，別に扱う．

サンプル

スクリーン座標（１００，５０）に黄色の点を打つ．画面モードはカラーで６４０×４００とする．

```
MOV  AX, 0060H
MOV  DS, AX
MOV  BX, 640H
MOV  CH, 0B0H
MOV  BYTE  [BX], 6              ;黄色
MOV  WORD  [BX+8], 64H          ;X=100
MOV  WORD  [BX+0AH], 32H;Y=50
MOV  WORD  [BX+0CH], 1          ;長さ=1
MOV  WORD  [BX+0EH], 80H;オフセット
PUSH  CS
POP  ES                         ;セグメント
MOV  AH, 45H
INT  18H
        .
ORG  80H
DB  0FFH
```

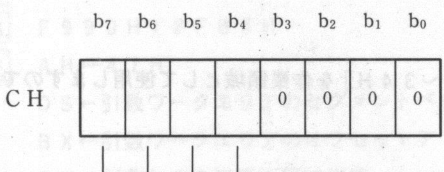
2．7　グラフィックスクリーンからの ドット列の読み出し

機　能	グラフィックスクリーンからドット列を読み出します.
割　込	ＩＮＴ　１８Ｈ
絶対番地	Ｆ９９０Ｈ：２ＥＥＡＨ
入力条件	ＡＨ←４６Ｈ

ＤＳ←引数ワークエリアのセグメントベース

ＢＸ←引数ワークエリアのオフセットアドレス

ＣＨ←対象とする描画画面の指定



描画画面ナンバ

0　0：プレーン１

0　1：プレーン２

1　0：プレーン３

1　1：プレーン１～３

（３画面同時読み出し）

描画画面範囲

0：ＬＯＷＥＲまたはＡＬＬ

1：ＵＰＰＥＲ

解像度モード

0：６４０×２００

1：６４０×４００

ＥＳ←ドットパターン・バッファのセグメントベース

［ＢＸ＋８～９］←ドット列の読み出しを始めるスクリーンＸ座標

（０～６３９の範囲にあること）

［ＢＸ＋０ＡＨ～０ＢＨ］←ドット列の読み出しを始めるスクリーン

Ｙ座標　（０～１９９ or ３９９）

［ＢＸ＋０ＣＨ～０ＤＨ］←読み出すドット列の長さ（ドット数）

［ＢＸ＋１０Ｈ～１１Ｈ］←読み出したドットパターンを格納するパターンバッファの先頭オフセットアドレス単一プレーンの場合または，３画面同時の場合のプレーン１用

［ＢＸ＋１２Ｈ～１３Ｈ］←プレーン２から読み出したドットパターンを格納するパターンバッファの先頭オフセットアドレス，３画面同時の場合のみ

［ＢＸ＋１４Ｈ～１５Ｈ］←プレーン３から読み出したドットパターンを格納するパターンバッファの先頭オフセットアドレス，３画面同時の場合のみ

出力条件　なし

破　壊　なし

作業領域　［ＢＸ＋２ＡＨ～３４Ｈ］を作業領域として使用しますので，確保して下さい．

解　説

［ＢＸ＋８～９］と［ＢＸ＋０ＡＨ～０ＢＨ］で示されるＸ，Ｙ座標を始点として，長さ［ＢＸ＋０ＣＨ～０ＤＨ］のドット列をパターンバッファに読み込みます．

単一プレーンのみを読み込むとき(ＣＨのb_5，b_4が１，１以外のとき)のパターンバッファのアドレスは，［ＢＸ＋１０～１１］に与えます．３プレーン同時読み込みのとき(ＣＨのb_5，b_4が１，１のとき)は，プレーン毎に対応するパターンバッファを指定します．

処　理

本サブルーチンでは，ＧＤＣを使用せず，ＣＰＵから直接グラフィックＶＲＡＭ（０Ａ８０００Ｈ～０ＢＦＦＦＦＨ）をＲＥＡＤしています．

(1) ＧＤＣがＶＲＡＭに書き込み中のときはそれが終了するまで待つ．

```
WAIT1: IN    AL, 0A0H     FIFOが空になる
       TEST  AL, 04       まで待つ．
       JE    WAIT1
WAIT2: IN    AL, 0A0H     描画中フラグが降り
       TEST  AL, 08       るまで待つ．
       JNE   WAIT2
```

(2) X，Y座標からVRAM上の読み込み開始アドレスを計算する．

ADRS＝X／8＋Y＊80

(3) 転送は先頭を揃えてバイト単位で行う．読み込み長が8の倍数でないときは最後のバイトの余りビットは不定となる．

2．8　直線および四角形の描画

機　能	指定された描画画面に直線または四角形を書き込む．
割　込	INT 18H
絶対番地	F990H：2FB7H
入力条件	AH←47H
	DS←引数ワークエリアのセグメントベース
	BX←引数ワークエリアのオフセットアドレス
	CH←対象とする描画画面の指定

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
CH					0	0	0	0

b_5 b_4：描画画面ナンバ

0　0：プレーン1

0　1：プレーン2

1　0：プレーン3

1　1：プレーン1～3

（3画面同時書き込み）

b_6：描画画面範囲

0：LOWERまたはALL

1：UPPER

b_7：解像度モード

0：640×200

1：640×400

ＥＳ←ドットパターン・バッファのセグメントベース

［ＢＸ＋０］←３画面同時書き込みの場合の描画モード

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
［ＢＸ＋０］	0	0	0	0	0			

b_0 ── プレーン１
b_1 ── プレーン２
b_2 ── プレーン３

各プレーンに対して
クリアまたはセットを
選択できる．

0：CLEAR
1：SET

［ＢＸ＋２］←単一画面書き込みの場合の描画モード

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
［ＢＸ＋２］	0	0	0	0	0	0		

下位２ビットで４種類の
モードを選択できる．

b_1	b_0	
0	0	REPLACE
0	1	COMPLEMENT
1	0	CLEAR
1	1	SET

［ＢＸ＋３］←四角形の描画の場合，描画方向を０～７の数値で指定します．詳しくは次ページを御覧下さい．

［ＢＸ＋８～９］←描画開始点のＸ座標
（０～６３９の範囲にあること）

［ＢＸ＋０ＡＨ～０ＢＨ］←描画開始点のＹ座標
（０～１９９or３９９）

［ＢＸ＋１６Ｈ～１７Ｈ］←描画終了点のＸ座標
（０～６３９の範囲であること）

［ＢＸ＋１８Ｈ～１９Ｈ］←描画終了点のＹ座標
（０～１９９or３９９）

［ＢＸ＋２０Ｈ〜２１Ｈ］←ラインスタイル

１６ドットでラインスタイルを指定します．書き込む直線の長さが１６ドットを超える場合，このラインスタイルが１６ドット毎に繰返し使われます．

［ＢＸ＋２８Ｈ］←描画タイプ

１：直線

２：四角形

出力条件　なし

破　　壊　なし

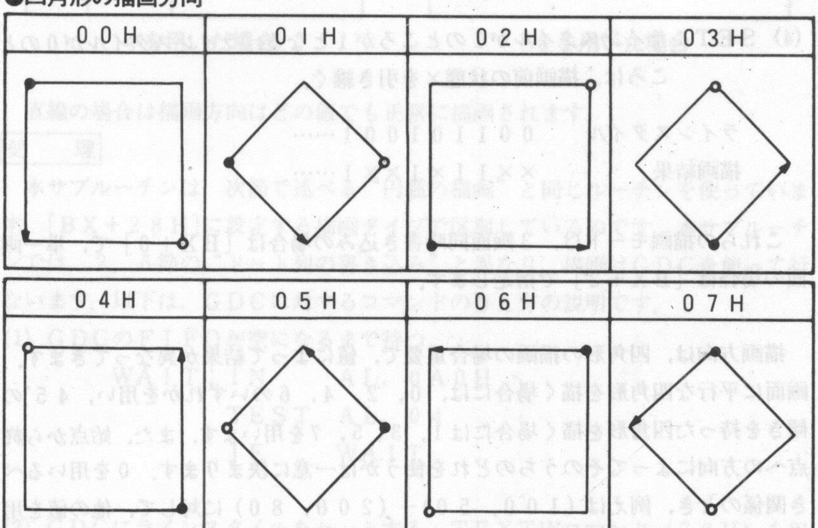
作業領域　［ＢＸ＋２ＡＨ〜４９Ｈ］を作業領域として使用しますので，確保して下さい．

解　　説

レジスタＣＨによって指定された描画画面に直線または四角形を書き込みます．描画は描画モードに従って１６ビットのラインスタイルと描画画面との演算結果を書き込むことによって行います．

３画面同時書き込み（ＣＨのb_5，b_4が１，１のとき）か否かで選択できる描画モ

●四角形の描画方向



●描画始点
○描画終点

ードが異なります．３画面同時の場合はＣＬＥＡＲとＳＥＴのみ，単一画面の場合はＲＥＰＬＡＣＥ，ＣＯＭＰＬＥＭＥＮＴ，ＣＬＥＡＲ，ＳＥＴの４種類を選択できます．以下にそれらの動作を説明します．

(1) ＲＥＰＬＡＣＥ：ラインスタイルをそのまま画面に書き込む．

ラインスタイル	００１１０１００１……
描画結果	００１１０１００１……

(2) ＣＯＭＰＬＥＭＥＮＴ：ラインスタイルと描画前の状態と排他的論理和（ＸＯＲ）をとって画面に書き込む．

描画前の状態	１１１００１０１０……
ラインスタイル	００１１０１００１……
描画結果	１１０１０００１１……

(3) ＣＬＥＡＲ：ラインスタイルが１のところが０となる．ラインスタイルが０のところは，描画前の状態×を引き継ぐ．

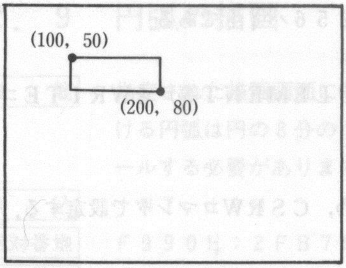
ラインスタイル	００１１０１００１……
描画結果	××００×０××０……

(4) ＳＥＴ：ラインスタイルが１のところが１となる．ラインスタイルが０のところは，描画前の状態×を引き継ぐ．

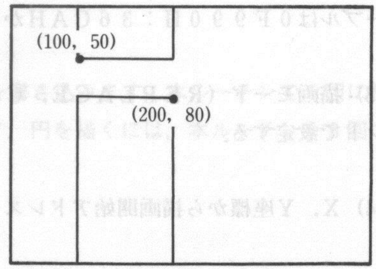
ラインスタイル	００１１０１００１……
描画結果	××１１×１××１……

これらの描画モードは，３画面同時書き込みの場合は［ＢＸ＋０］で，単一画面の場合は［ＢＸ＋２］で指定します．

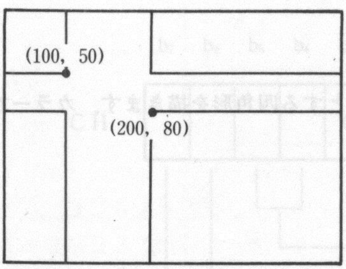
描画方向は，四角形の描画の場合重要で，値によって結果が異なってきます．画面に平行な四角形を描く場合には，０，２，４，６のいずれかを用い，４５°の傾きを持った四角形を描く場合には１，３，５，７を用います．また，始点から終点への方向によってそのうちのどれを使うかは一意に決まります．０を用いるべき関係のとき，例えば（１００，５０）－（２００，８０）に対して，他の値を用いるとどういう結果が得られるかを次に示します．



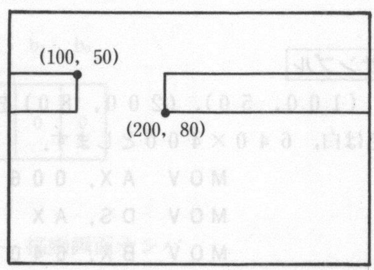
0を用いた場合



2を用いた場合



4を用いた場合



6を用いた場合

直線の場合は描画方向はどの値でも正常に描画されます．

処　理

本サブルーチンは，次節で述べる“円弧の描画”と同じルーチンを使っています．［BX＋28H］に設定する描画タイプで区別しているのです．本サブルーチンでは，2．6節の“ドット列の書き込み”と異なり，描画はGDCを使って行ないます．以下は，GDCに対するコマンドの与え方の説明です．

(1) GDCのFIFOが空になるまで待つ．

```
WAIT：IN    AL，0A0H
      TEST  AL，04
      JE    WAIT
```

(2) GDCにラインスタイルをセットする．TEXTWコマンド（78H）を用いる．第1章で説明したようにGDCを介するとビット順が逆になるので，ラインスタイルの設定ではあらかじめテーブルを使ってビットを逆順にしている．テ

ーブルは０Ｆ９９０Ｈ：３６ＣＡＨからの２５６バイトである．

(3) 描画モード（ＲＥＰＬＡＣＥ，ＣＯＭＰＬＥＭＥＮＴ等）をＷＲＩＴＥコマンドで設定する．

(4) Ｘ，Ｙ座標から描画開始アドレスを求め，ＣＳＲＷコマンドで設定する．

(5) 描画方向，描画種類（直線，四角形等），その他の描画に必要なパラメータをＶＥＣＴＷコマンドで設定する．

(6) ＶＥＣＴＥコマンドで描画の実行を指示する．

サンプル

（１００，５０），（２００，８０）を頂点とする四角形を描きます．カラーで，色は白，６４０×４００とします．

```
MOV  AX, 0060H
MOV  DS, AX
MOV  BX, 640H
MOV  CH, 0B0H
MOV  BYTE  [BX], 7              ;白
MOV  BYTE  [BX+3], 0            ;描画方向
MOV  WORD  [BX+8], 64H          ;X=100
MOV  WORD  [BX+0AH], 32H        ;Y=50
MOV  WORD  [BX+16H], 0C8H       ;X2=200
MOV  WORD  [BX+18H], 50H        ;Y2=80
MOV  WORD  [BX+20H], 80H        ;ラインスタイル
PUSH  CS                        ;オフセット
POP  ES                         ;セグメント
MOV  BYTE  [BX+28H], 2          ;四角形
MOV  AH, 47H
INT  18H
```

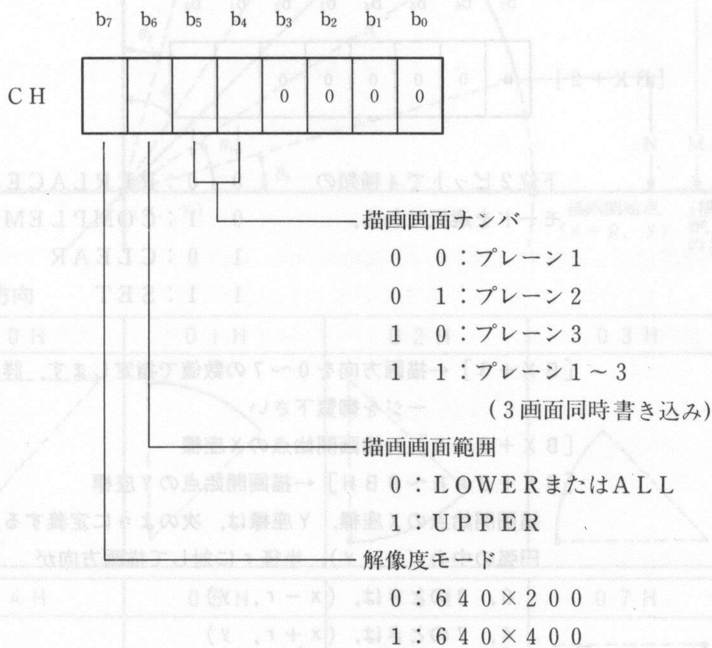
2．9　円弧の描画

機　　能	指定された描画画面に円弧を書き込みます．本ルーチンで１回に描ける円弧は円の８分の１です．円を描くには，本ルーチンを８回コールする必要があります．
割　　込	ＩＮＴ　１８Ｈ
絶対番地	Ｆ９９０Ｈ：２ＦＢ７Ｈ
入力条件	ＡＨ←４８Ｈ

ＤＳ←引数ワークエリアのセグメントベース

ＢＸ←引数ワークエリアのオフセットアドレス

ＣＨ←対象とする描画画面の指定



ＥＳ←ドットパターン・バッファのセグメントベース

［ＢＸ＋０］←３画面同時書き込みの場合の描画モード

（カラーの場合はパレット番号を意味する）

	b7	b6	b5	b4	b3	b2	b1	b0
［ＢＸ＋０］	0	0	0	0	0			

b0 → プレーン1
b1 → プレーン2
b2 → プレーン3

各プレーンに対して
クリアまたはセットを
選択できる．

0：CLEAR
1：SET

［ＢＸ＋２］←単一画面書き込みの場合の描画モード

	b7	b6	b5	b4	b3	b2	b1	b0
［ＢＸ＋２］	0	0	0	0	0	0		

下位２ビットで４種類の
モードを選択できる．

0　0：RERLACE
0　1：COMPLEMENT
1　0：CLEAR
1　1：SET

［ＢＸ＋３］←描画方向を０～７の数値で指定します．詳しくは次ページを御覧下さい．

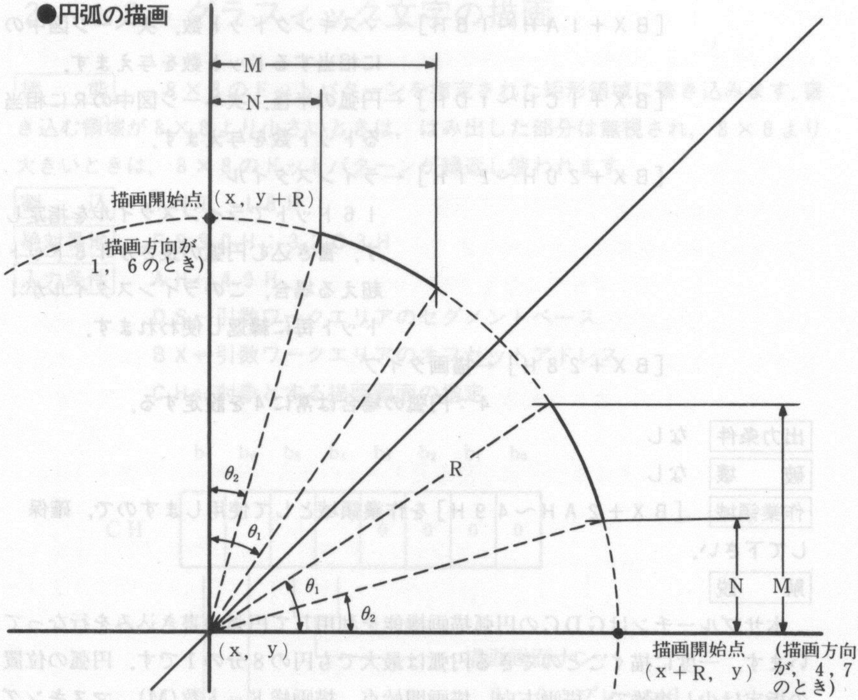
［ＢＸ＋８～９］←描画開始点のＸ座標

［ＢＸ＋０ＡＨ～０ＢＨ］←描画開始点のＹ座標

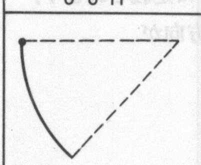
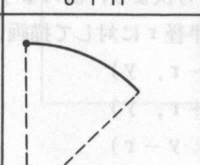
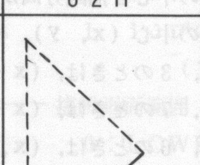
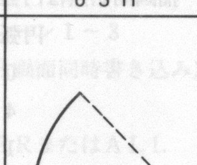
描画開始点のＸ座標，Ｙ座標は，次のように定義する．
円弧の中心（ｘ，ｙ），半径ｒに対して描画方向が
０，３のときは，（ｘ－ｒ，ｙ）
４，７のときは，（ｘ＋ｒ，ｙ）
Ｉ，６のときは，（ｘ，ｙ－ｒ）
２，５のときは，（ｘ，ｙ＋ｒ）．

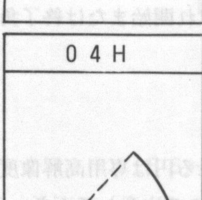
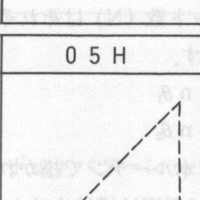
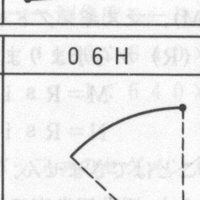
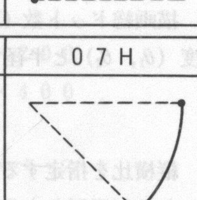
［ＢＸ＋０ＣＨ～０ＤＨ］←描画総ドット数．次ページ図中のＭに相当するドット数を与えます．

●円弧の描画



●描画方向

00H	01H	02H	03H
			

04H	05H	06H	07H
			

［ＢＸ＋１ＡＨ〜１ＢＨ］←マスキングドット数．次ページ図中のＮに相当するドット数を与えます．

［ＢＸ＋１ＣＨ〜１ＤＨ］←円弧の半径．次ページ図中のＲに相当するドット数を与えます．

［ＢＸ＋２０Ｈ〜２１Ｈ］←ラインスタイル

１６ドットでラインスタイルを指定します．書き込む円弧の長さが１６ドットを超える場合，このラインスタイルが１６ドット毎に繰返し使われます．

［ＢＸ＋２８Ｈ］←描画タイプ

４：円弧の場合は常に４を設定する．

出力条件　なし

破　　壊　なし

作業領域　［ＢＸ＋２ＡＨ〜４９Ｈ］を作業領域として使用しますので，確保して下さい．

解　　説

本サブルーチンはＧＤＣの円弧描画機能を利用して円弧の書き込みを行なっています．一度に描くことのできる円弧は最大でも円の８分の１です．円弧の位置の指定は少し複雑で，描画方向，描画開始点，描画総ドット数(M)，マスキングドット数（N)，半径（R）を指定することによって行います．

描画開始点は円弧の中心と描画方向から決まり，次のように定義されます．

円弧の中心（x，y)，半径ｒに対して描画方向が

0，3のときは，(x－r，y)

4，7のときは，(x＋r，y)

1，6のときは，(x，y－r)

2，5のときは，(x，y＋r)

描画総ドット数(M)，マスキングドット数（N）はそれぞれ開始または終了角度（θ_1，θ_2）と半径（R）から求まります．

$$M = R\sin\theta_1$$

$$N = R\sin\theta_2$$

縦横比を指定することはできません．本ルーチンで描かれる円は専用高解像度のときに正円となります．標準解像度では正円は描けませんので注意して下さい．

対象とする描画画面の指定，描画モード，ラインスタイルに関しては，２．８の直線，四角形の描画と同じです．

2．10 グラフィック文字の描画

機　能	8×8のドットパターンを指定された矩形領域に書き込みます．書き込む領域が8×8より小さいときは，はみ出した部分は無視され，8×8より大きいときは，8×8のドットパターンが繰返し使われます．
割　込	INT 18H
絶対番地	F990H：32C3H
入力条件	AH←49H

DS←引数ワークエリアのセグメントベース

BX←引数ワークエリアのオフセットアドレス

CH←対象とする描画画面の指定

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
CH					0	0	0	0

b_5, b_4 — 描画画面ナンバ

0 0：プレーン1

0 1：プレーン2

1 0：プレーン3

1 1：プレーン1～3

（3画面同時書き込み）

b_6 — 描画画面範囲

0：LOWERまたはALL

1：UPPER

b_7 — 解像度モード

0：640×200

1：640×400

ES←ドットパターン・バッファのセグメントベース

［ＢＸ＋０］←３画面同時書き込みの場合の描画モード

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
［ＢＸ＋０］	0	0	0	0	0			

各プレーンに対して
クリアまたはセットを
選択できる．

b_0 ―プレーン１
b_1 ―プレーン２
b_2 ―プレーン３

０：ＣＬＥＡＲ
１：ＳＥＴ

［ＢＸ＋２］←単一画面書き込みの場合の描画モード

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
［ＢＸ＋２］	0	0	0	0	0	0		

下位２ビットで４種類の
モードを選択できる．

０　０：ＲＥＲＬＡＣＥ
０　１：ＣＯＭＰＬＥＭＥＮＴ
１　０：ＣＬＥＡＲ
１　１：ＳＥＴ

［ＢＸ＋３］←描画方向を０～７の数値で指定します．詳しくは次のページを御覧下さい．

［ＢＸ＋８～９］←描画領域の左下のＸ座標

［ＢＸ＋０ＡＨ～０ＢＨ］←描画領域の左下のＹ座標

［ＢＸ＋０ＣＨ～０ＤＨ］←描画領域の第１描画方向ドット数．ただし，描画領域が８×８の場合は，０を与える．

［ＢＸ＋１ＥＨ～１ＦＨ］←描画領域の第２描画方向ドット数－１．ただし，描画領域が８×８の場合は，０を与える．

［ＢＸ＋２０Ｈ～２７Ｈ］←８×８のドットパターン

出力条件　なし

破　　壊　　なし

作業領域　　［ＢＸ＋２ＡＨ～３２Ｈ］を作業領域として使用しますので，確保して下さい．

解　　説

ＧＤＣのグラフィック文字の描画機能を利用してドットパターンの書き込みを行ないます．ドットパターンは８ドット×８ドットで与え，描画領域がそれより小さいときは，はみ出した部分が無視され，また，大きい場合は，同じパターンを繰り返し使って，描画領域全体を塗りつぶします．

描画方向で指定する数値によってドットパターンを描く角度を変えることができます．直線，円弧などと同様に書き込みにあたっては４種類の描画モードを選択できます．各モードの動作については２．８の解説を参照して下さい．

処　　理

ＧＤＣを使って描画を行っています．以下はその手順です．

(1) ＴＥＸＴＷコマンドで，８×８のドットパターンをＧＤＣに設定する．

(2) ＷＲＩＴＥコマンドで，描画モードを設定する．

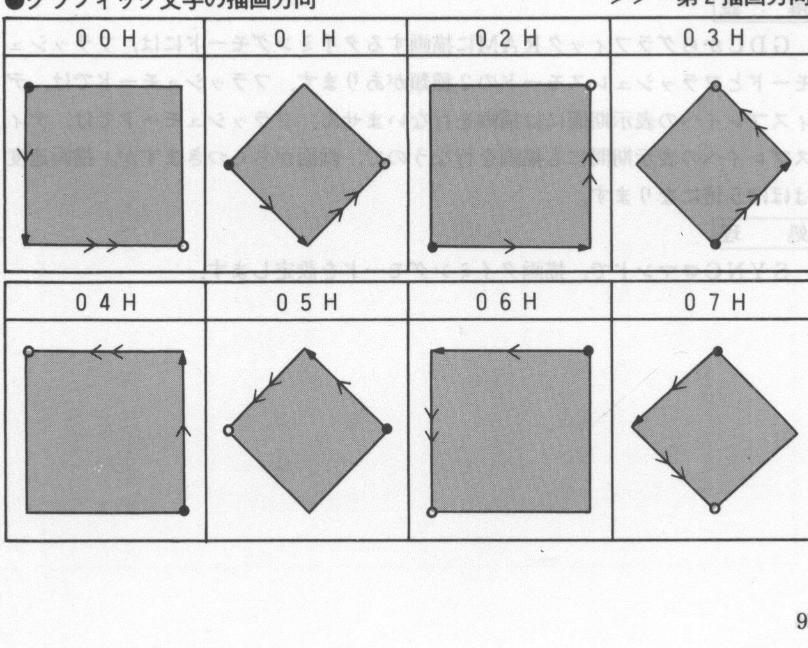
(3) ＣＳＲＷコマンドで，描画領域を設定する．

(4) ＶＥＣＴＷコマンドで，描画方向を設定する．

(5) ＴＥＸＴＥコマンドで，描画の実行を開始する．

──＞──第１描画方向
──＞＞──第２描画方向

●グラフィック文字の描画方向



2．11 高速書き込みモードの選択

機　能　　GDCから描画するときのタイミングをフラッシュモード（高速書き込み）にするか，フラッシュレスモードにするかを選択します．

割　込　　INT 18H

絶対番地　　F990H：3467H

入力条件　　AH←4AH

CH←描画タイミングモードの設定

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
CH	0	0	0		0	1	1	0

b_4 ─描画タイミングモード
0：フラッシュ描画（高速書き込み）
1：フラッシュレス描画

出力条件　　なし

破　壊　　なし

解　説

GDCからグラフィックRAMに描画するタイミングモードには，フラッシュモードとフラッシュレスモードの2種類があります．フラッシュモードでは，ディスプレイへの表示期間には描画を行ないません．フラッシュモードでは，ディスプレイへの表示期間にも描画を行なうので，画面がちらつきますが，描画速度はほぼ5倍になります．

処　理

SYNCコマンドで，描画タイミングモードを設定します．

第3章

グラフィックLIO

2.11 高速書き込みモードの選択

機能 GDCから描画するときのタイミングをフラッシュモード(高速書き込み)にするか、フラッシュレスモードにするかを選択します。

割込 INT 18H

絶対番地 F990H：3467H

入力条件 AH←4AH

CH←描画タイミングモードの設定

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
CH	0	0	0		0	1	1	0

b_4：描画タイミングモード

0：フラッシュ描画（高速書き込み）

1：フラッシュレス描画

出力条件 なし

破壊 なし

解説

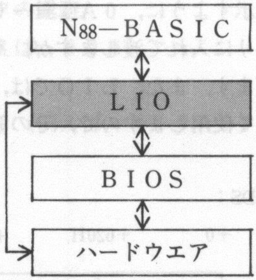
GDCからグラフィックRAMに描画するタイミングモードには、フラッシュモードとフラッシュレスモードの2種類があります。フラッシュモードでは、ディスプレイへの表示期間には描画を行ないません。フラッシュモードでは、ディスプレイへの表示期間にも描画を行なうので、画面がちらつきますが、描画速度はほぼ5倍になります。

処理

SYNCコマンドで、描画タイミングモードを設定します。

3．0　ＬＩＯの概要

第０章で説明したように，ＰＣ−９８００シリーズのソフトウェア構造は階層的になっており，ＢＡＳＩＣのすぐ下に位置し，ＢＡＳＩＣコマンドの下請け作業をするところが，ＬＩＯ (Logical Input Output)と呼ばれるサブルーチンの集まりです．グラフィックＬＩＯの各サブルーチンは１７種類あり，ＢＡＳＩＣのコマンドにほぼ１対１に対応しています．マシン語で，ＢＡＳＩＣのグラフィック機能を実現したい場合には，ＬＩＯを使用するのが最も便利でしょう．



ＬＩＯの位置付け

ＬＩＯルーチンと対応するＢＡＳＩＣコマンド

割込番号	ＬＩＯ	対応するＢＡＳＩＣコマンド
ＡＯ	ＧＩＮＩＴ	なし
Ａ１	ＧＳＣＲＥＥＮ	ＳＣＲＥＥＮ
Ａ２	ＧＶＩＥＷ	ＶＩＥＷ
Ａ３	ＧＣＯＬＯＲ１	ＣＯＲＯＲ
Ａ４	ＧＣＯＬＯＲ２	ＣＯＬＯＲ＝（　，　）
Ａ５	ＧＣＬＳ	ＣＬＳ　２
Ａ６	ＧＰＳＥＴ	ＰＳＥＴ，ＰＲＥＳＥＴ
Ａ７	ＧＬＩＮＥ	ＬＩＮＥ
Ａ８	ＧＣＩＲＣＬＥ	ＣＩＲＣＬＥ
Ａ９	ＧＰＡＩＮＴ１	ＰＡＩＮＴ（単色）
ＡＡ	ＧＰＡＩＮＴ２	ＰＡＩＮＴ（タイル）
ＡＢ	ＧＧＥＴ	ＧＥＴ
ＡＣ	ＧＰＵＴ１	ＰＵＴ（グラフィック）
ＡＤ	ＧＰＵＴ２	ＰＵＴ（漢字）
ＡＥ	ＧＲＯＬＬ	ＲＯＬＬ
ＡＦ	ＧＰＯＩＮＴ	？＝ＰＯＩＮＴ（　，　）
ＣＥ	ＧＣＯＰＹ	なし

□ＬＩＯの呼び出し方法

ＩＮＴ命令によるソフトウェア割込を使って呼び出します．割込番号は，表に示すように，０Ａ１Ｈ～０ＡＦＨと０ＣＥＨです．引数はほとんどの場合，メモリに入れて渡しますが，その先頭アドレスをＤＳ：ＢＸレジスタが指すようにします．また，ＬＩＯでは，データセグメント上の特定の領域をワークエリアとして使用しますので，その領域は確保しておかなくてはなりません．

DS：

+0　　+620H　　+6A0H　　+1000H　　+1200H　　+1380H　　+1400H

未使用	WORK 128B	未使用	WORK 512B	未使用	GCOPY用 WORK 128B

グラフィックＬＩＯワークエリア

ワークエリアは，データセグメント上の３つの領域に分散しています．必ず確保しなくてはならないのは，ＤＳ：［６２０Ｈ～６９ＦＨ］およびＤＳ：［１０００Ｈ～１１ＦＦＨ］で，ＧＣＯＰＹを使用する場合は，さらにＤＳ：［１３８０Ｈ～１３ＦＦＨ］が必要となります．

〈N88－ＢＡＳＩＣからＬＩＯを使う場合〉

ＤＳ＝６０Ｈ　としてください．既に領域が確保されていますので，ワークエリアに気を使う必要はありません．引数エリアとしては，１５０ＡＨからの領域を使うことができます．

〈ＣＰ／ＭまたはＭＳ－ＤＯＳからＬＩＯを使う場合〉

１４００Ｈバイト（ＧＣＯＰＹを使わない場合は１２００Ｈバイト）の領域を確保してその先頭アドレスをＤＳに入れて下さい．引数エリアもデータセグメント上に確保して下さい．図中の未使用領域を使うこともできます．

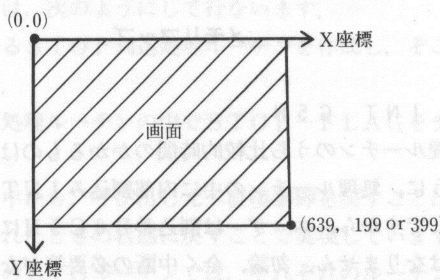
レジスタは，ＤＳ，ＳＳ，ＳＰを除いて破壊されます．必要なレジスタは，呼び出す前に退避しておいてください．ＬＩＯから戻ったときの終了条件はレジスタＡＨにセットされています．ＡＨが０なら正常終了，０以外なら異常終了です．

一般的な呼び出しのコーディングは次のようになります．

```
PUSH レジスタs      必要なレジスタの退避
MOV  AX，データセグメント  （例えば，60H）
MOV  DS，AX
MOV  BX，引数エリア先頭アドレス  （例えば，150AH）
INT  割込番号
CMP  AM，0      終了条件件チェック
POP  レジスタs  レジスタの復帰
JNE  エラー処理
          ・
          ・
```

□ＬＩＯの座標系

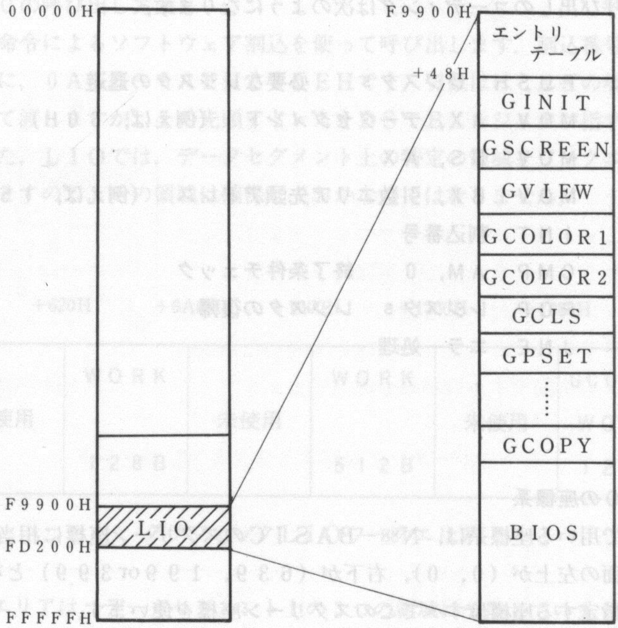
ＬＩＯで用いる座標系は，N88－ＢＡＳＩＣのスクリーン座標に相当するものです．画面の左上が（０，０），右下が（６３９，１９９or３９９）となります．ＬＩＯで指定する座標はすべてこのスクリーン座標を使います．



ＬＩＯの座標系

□メモリマップ

ＬＩＯの処理ルーチンは，ROM上の実アドレス０Ｆ９９００Ｈからにあります．先頭から４８Ｈバイトには，各処理ルーチンのエントリアドレスのテーブルがあり，システムの起動時にはこのテーブルを参照してＬＩＯの割込みベクトルテーブルが作られます．



メモリマップ

□内部割込　ＩＮＴ　Ｃ５Ｈ

ＬＩＯの処理ルーチンのうち比較的時間のかかるものは，途中で処理の中断が可能となるように，処理ルーチンの中に内部割込みＩＮＴ　Ｃ５Ｈ命令が埋め込まれています．ですから，ユーザーは割込番号０Ｃ５Ｈに対する処理ルーチンを用意しなくてはなりません．勿論，全く中断の必要等がない場合は，Ｃ５Ｈの割込ベクタがＩＲＥＴ命令を指すようにすればいいわけです．N88－ＢＡＳＩＣを起動させた状態では，Ｃ５Ｈの割込ベクタはＳＴＯＰキーによる中断処理のルーチンを指しています．このルーチンでは，ＳＴＯＰキーが押されていると，ただちにＢＡＳＩＣのコマンド待ち状態に戻るように作られています．ＢＡＳＩＣ上でＬＩＯを利用する場合で，かつ，ＳＴＯＰキーによる中断を望む場合は，このままでも使えますが，その他の場合は，必ずユーザが割込処理ルーチンを用意して下さい．次にこの割込処理ルーチンの作成例と設定方法について述べます．

［ＩＮＴ　Ｃ５Ｈ　割込処理ルーチン作成例］

このルーチンでは，ＳＴＯＰキーが押されていると描画処理を中断してＬＩＯの呼び出しへ戻るという処理をします．

```
0000 50             PUSH    AX    ┐
0001 53             PUSH    BX    │
0002 51             PUSH    CX    │
0003 52             PUSH    DX    │
0004 55             PUSH    BP    │
0005 56             PUSH    SI    │
0006 1E             PUSH    DS    │
0007 06             PUSH    ES    │
0008 31C0           XOR     AX,AX │
000A 8EC0           MOV     ES,AX │  ＳＴＯＰキーが押されたかどうかを
000C CD83           INT     83    │  調べる。
000E F7064A040080   TEST    WORD [044A],8000
0014 07             POP     ES    │  押された場合…Ｚフラグリセット
0015 1F             POP     DS    │  押れれない場合…Ｚフラグセット
0016 5E             POP     SI    │
0017 5D             POP     BP    │
0018 5A             POP     DX    │
0019 59             POP     CX    │
001A 5B             POP     BX    │
001B 58             POP     AX    ┘
001C 740A           JE      0028 ---------------押されない場合は２８Ｈへ
001E 8B260410       MOV     SP,[1004] ┐ ＳＰとＢＸをＬＩＯがコールされたときの
0022 8B1E0610       MOV     BX,[1006] ┘ 状態に戻す。
0026 B400           MOV     AH,00
0028 CF             IRET
```

ＢＡＳＩＣでは，ＳＴＯＰキー押下のチェックはＩＮＴ　８３Ｈを使って行なえます．他のＯＳ上では，次のようにして行ないます．

・割込番号０６に対するＳＴＯＰ割込処理ルーチンを作成し，そこでＳＴＯＰ－ＦＬＡＧをセットする．

・割込番号０Ｃ５Ｈの処理ルーチンの中でＳＴＯＰ－ＦＬＡＧをチェックする．

割込処理ルーチンの中から，呼び出し元へ直接制御を戻すことは，スタックポインタの値をコールされたときの状態に戻すことで実現しています．ＧＣＩＲＣＬＥ，ＧＰＡＩＮＴ等のＬＩＯルーチンでは，それぞれのルーチンの先頭でスタックポインタの値を退避しているのでそれを使っています．

割込処理ルーチンが完成したら，その先頭アドレスを割込ベクタ・テーブルにセットします．ＩＮＴ　Ｃ５Ｈに対応する割込ベクタ・テーブルは００００Ｈ：［０３１４Ｈ～０３１７Ｈ］です．

3．ＩＧＩＮＩＴ（グラフィックＬＩＯの初期化）

機能	グラフィックＬＩＯの初期化を行ないます．
絶対番地	Ｆ９９０Ｈ：００４８Ｈ
割込	ＩＮＴ　Ａ０Ｈ
入力条件	ＤＳ←グラフィックＬＩＯワークエリアのセグメントベース（通常００６０Ｈ）
引数	なし
出力条件	ＡＨ←０
破壊	ＤＳ，ＳＳ，ＳＰを除くすべて
解説	

グラフィックＬＩＯを使用する場合は，最初に一度だけＧＩＮＩＴを実行しなければなりません．ＧＩＮＩＴでは次の処理を行っています．

(1) パレット番号の初期化（0～7を黒，青，赤，紫，緑，水色，黄，白に対応させる．）

(2) 画面モード←0（カラーグラフィック　６４０×２００）

(3) 画面スイッチ←0（グラフィック表示ＯＮ，高速書き込みなし）

(4) アクティブ画面←0（プレーン１を選択）

(5) ディスプレイ画面←1（プレーン１を選択）

(6) バックグラウンドカラー←0（黒を選択）

(7) ボーダーカラー←0（黒を選択）

(8) フォアグランドカラー←7（白を選択）

N88－ＢＡＳＩＣでは，起動時にＧＩＮＩＴを実行しています．

処理

ＧＩＮＩＴでは，まず，ＤＳ：［６２６Ｈ～６２ＤＨ］のカラーパレットの格納エリアにそれぞれ　0～7をストアし，ＧＳＣＲＥＥＮ（ＩＮＴ　Ａ1Ｈ）及びＧＣＯＬＯＲ1（ＩＮＴ　Ａ3Ｈ）を呼んでいます．

サンプル

グラフィックＬＩＯの初期化を行います．

```
MOV  AX, 0060H
MOV  DS, AX
INT  0A0H
```

３．２　ＧＳＣＲＥＥＮ
（クラフィック画面に対するモード設定）

機能	画面モード，画面スイッチ，アクティブ画面，ディスプレイ画面を設定する．
絶対番地	Ｆ９９０Ｈ：００８７Ｈ
割込	ＩＮＴ　ＡＩＨ
入力条件	ＤＳ←グラフィックＬＩＯワークエリアのセグメントベース ＢＸ←引数リストの先頭オフセットアドレス(データセグメント内)
引数	［ＢＸ＋０］←画面モード ［ＢＸ＋Ｉ］←画面スイッチ ［ＢＸ＋２］←アクティブ画面 ［ＢＸ＋３］←ディスプレイ画面

DS：BX+0	+1	+2	+3
画面 モード	画面 スイッチ	アクティ ブ画面	ディスプ レイ画面

出力条件	ＡＨ←終了条件（０なら正常終了）
破壊	ＤＳ，ＳＳ，ＳＰを除くすべて
解説	

Ｎ88−ＢＡＳＩＣのＳＣＲＥＥＮステートメントに対応しています．各引数の意味もＢＡＳＩＣと同じです．ただし，省略は０ＦＦＨを代入することで表わします．各引数の詳細を次に示します．

引数詳細

(1)画面モード

引数	画面モード	分解能	色
00H	カラーモード	640×200	カラー
01H	白黒モード	640×200	白黒
02H	高分解能白黒モード	640×400	白黒
03H	高分解能カラーモード	640×400	カラー
FFH	今までの状態を引継ぐ		

(2)画面スイッチ

引数	グラフィック表示	高速描画
00H	ON	なし
10H	ON	あり
02H	OFF	なし
03H	OFF	あり
FFH	今までの状態を引継ぐ	

③　アクティブ画面

引　数	画面モードごとのアクティブ・ページ指定値			
	0	1	2	3
1	0	0	0	0
2	1	1	1	1※
3	2※	2	2	×
4	3※	3	3※	×
5	×	4	4※	×
6	×	5	5※	×
7	×	6※	×	×
8	×	7※	×	×
9	×	8※	×	×
10	×	9※	×	×
11	×	10※	×	×
12	×	11※	×	×

×印：指定不可

※ＰＣ－９８０１では指定不可

④ディスプレイ画面

引数	画面モードごとの意味			
	0	1	2	3
0	全ページ表示しない	全ページ表示しない	全ページ表示しない	全ページ表示しない
1	ページ1のみ表示	ページ1のみ表示	ページ1のみ表示	ページ1のみ表示
2	ページ2のみ表示	ページ2のみ表示	ページ2のみ表示	×
3	×	ページ1,2を合成表示	ページ1,2を合成表示	×
4	×	ページ3のみ表示	ページ3のみ表示	×
5	×	ページ1,3を合成表示	ページ1,3を合成表示	×
6	×	ページ2,3を合成表示	ページ2,3を合成表示	×
7	×	ページ1,2,3を合成表示	ページ1,2,3を合成表示	×
8	全ページ表示しない	全ページ表示しない	全ページ表示しない	全ページ表示しない
9	×	ページ4のみ表示	×	×
10	×	ページ5のみ表示	×	×

11	×	ページ4,5を合成表示	×	×
12	×	ページ6のみ表示	×	×
13	×	ページ4,6を合成表示	×	×
14	×	ページ5,6を合成表示	×	×
15	×	ページ4,5,6を合成表示	×	×
16	全ページ表示しない	全ページ表示しない	全ページ表示しない	全ページ表示しない
17	ページ3のみ表示	ページ7のみ表示	ページ4のみ表示	ページ2のみ表示
18	ページ4のみ表示	ページ8のみ表示	ページ5のみ表示	×
19	×	ページ7,8を合成表示	ページ4,5を合成表示	×
20	×	ページ9のみ表示	ページ6のみ表示	×
21	×	ページ7,9を合成表示	ページ4,6を合成表示	×
22	×	ページ8,9を合成表示	ページ5,6を合成表示	×
23	×	ページ7,8,9を合成表示	ページ4,5,6を合成表示	×
24	全ページ表示しない	全ページ表示しない	全ページ表示しない	全ページ表示しない
25	×	ページ10のみ表示	×	×
26	×	ページ11のみ表示	×	×
27	×	ページ10,11を合成表示	×	×
28	×	ページ12のみ表示	×	×
29	×	ページ10,12を合成表示	×	×
30	×	ページ11,12を合成表示	×	×
31	×	ページ10,11,12を合成表示	×	×

（16〜31：9801では指定不可）

引数がＦＦＨのときは今までの状態を引継ぐ。

3．3　ＧＶＩＥＷ（描画領域の指定）

機　能　アクティブ画面内の描画領域（ビューポート）を指定します．また，ビューポート内の塗りつぶし，外枠の描画も行ないます．以後，図形描画は，ビューポート内にのみ行なわれます．

絶対番地　Ｆ９９０Ｈ：０３７ＢＨ

割　込　ＩＮＴ　Ａ２Ｈ

入力条件　ＤＳ←グラフィックＬＩＯワークエリアのセグメントベース
ＢＸ←引数リストの先頭オフセットアドレス(データセグメント内)

引　数　［ＢＸ＋０～１］←左上のＸ座標　（Ｘ１）
［ＢＸ＋２～３］←左上のＹ座標　（Ｙ１）
［ＢＸ＋４～５］←右下のＸ座標　（Ｘ２）
［ＢＸ＋６～７］←右下のＹ座標　（Ｙ２）
［ＢＸ＋８］←領域色　（省略の場合は，ＦＦＨを指定）
［ＢＸ＋９］←境界色　（省略の場合は，ＦＦＨを指定）

DS：BX+0	+2	+4	+6	+8	+9
Ｘ１	Ｙ１	Ｘ２	Ｙ２	領域色	境界色

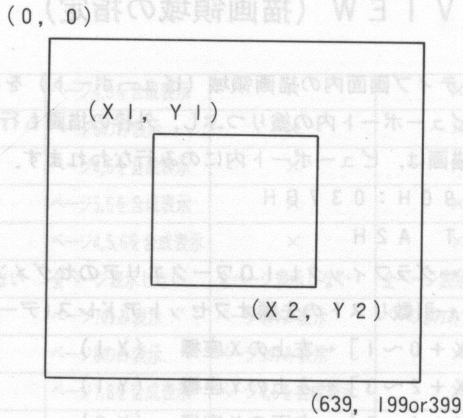
注意事項　引数は次の関係を満たすこと．
(1)　０≦Ｘ１＜Ｘ２≦３９９
(2)　０≦Ｙ１＜Ｙ２≦１９９（または　３９９　高分解能の時）

出力条件　ＡＨ←０　（正常終了）
ＡＨ←５　（引数の誤り）

破　壊　ＤＳ，ＳＳ，ＳＰを除くすべて

解　説

ＢＡＳＩＣのＶＩＥＷステートメントに対応しています．領域色，境界色の指定を省略すると，領域内の塗りつぶし，外枠の描画は行ないません．領域色，境界色の指定は，パレット番号で行ないます．



処　理

(1)　[BX+8]がFFHでない場合，矩形領域の塗りつぶしルーチンをコールする.（CALL　349FH)

(2)　[BX+9]がFFHでない場合，外枠の描画を行なう.これは，BIOSの直線描画ルーチンを4回コールすることで実現している.

(3)　ビューポート座標の格納領域DS：[062EH～0635H]への設定を行なう.

サンプル

ビューポートを（100，50)－（500，150）にセットし，外枠を赤で，内部を青で塗りつぶす.

```
MOV  AX, 0060H
MOV  DS, AX
MOV  BX, 150AH
MOV  WORD  [BX], 64H    ;X1=100
MOV  WORD  [BX+2], 32H  ;Y1=50
MOV  WORD  [BX+4], 1F4H;X2=500
MOV  WORD  [BX+6], 96H  ;Y2=150
MOV  BYTE  [BX+8], 1    ;領域色=青
MOV  BYTE  [BX+9], 2    ;境界色=赤
INT  0A2H
```

3．4　GCOLOR1（各部の色を指定）

機　能　バックグラウンドカラー，ボーダーカラー，フォアグラウンドカラーを指定します．

絶対番地　F990H：050CH

割　込　INT　A3H

入力条件　DS←グラフィックLIOワークエリアのセグメントベース
BX←引数リストの先頭オフセットアドレス(データセグメント内)

引　数　[BX＋1]←バックグラウンドカラー（パレット番号で指定）
[BX＋2]←ボーダーカラー（カラーコードで指定）
[BX＋3]←フォアグランドカラー（パレット番号で指定）
FFHを指定すると今までの状態を引継ぐ．

DS：BX+0	+1	+2	+3
未使用	バックグラウンドカラー	ボーダーカラー	フォアグラウンドカラー

出力条件　AH←0

破　壊　AX，CL，BP

解　説

BASICの[1] GCOLORステートメントに対応しています．ただし，ファンクションコードの設定はGCOLOR1ではできません．

〈バックグラウンドカラー〉とは，地の色のことで，GCLSで消去したときは，この色で塗り変えられます．

〈ボーダーカラー〉とは，ディスプレイの表示可能範囲外の色のことです．専用高解像度ディスプレイ使用時は，ボーダーカラーを指定しても色を付けることができません．ボーダーカラーのみ，パレット番号ではなく，直接カラーコードで指定します．

〈フォアグランドカラー〉とは，図形を描くときに使われる色のことです．

処　理

(1)　バックグランドカラーをワークエリアDS：［624H］に設定する.

(2)　ボーダーカラーをワークエリアDS：［641H］に設定し，BIOSボーダーカラーの設定をコールする.

(3)　フォアグランドカラーをワークエリアDS：［623H］に設定する.

サンプル

BASICのステートメント　COLOR，2，3，4　を機械語で実現します.

```
MOV  AX, 0060H
MOV  DS, AX
MOV  BX, 150AH
MOV  BYTE  [BX+1], 2 ; バックグラウンドカラー
MOV  BYTE  [BX+2], 3 ; ボーダーカラー
MOV  BYTE  [BX+3], 4 ; フォアグランドカラー
INT  0A3H
```

3．5　GCOLOR2（カラーパレットの変更）

機　能　パレット番号とカラーコードの対応付けを変更します．

絶対番地　F990H：054BH

割　込　INT　A4H

入力条件　DS←グラフィックLIOワークエリアのセグメントベース

BX←引数リストの先頭オフセットアドレス(データセグメント内)

引　数　[BX+0]←パレット番号

[BX+1]←カラーコード

DS：BX+0　+1

パレット番号	カラーコード

注意事項　パレット番号，カラーコードは，0～7の範囲を厳守すること．このサ　ブルーチンでは，引数のチェックをしていないので，ワーク領域を壊す恐れがあります．

入力条件　AH←0

破　壊　AX，CL，SI，DI，ES

解　説

BASICのステートメント　COLOR＝（パレット番号，カラーコード）に対応しています．

処　理

(1)　ワークエリアDS：［626H～］を書き換える．

(2)　BIOSカラーパレットの設定をコールする．

サンプル

カラーパレットを初期状態に戻すBASICのプログラムを機械語でLIOを使って機械語で実現します．

［BASICプログラム］

```
10 FOR I=0 TO 7
20    COLOR=(I, I)
```

30 NEXT I

［機械語プログラム］

```
            MOV  AX, 0060H
            MOV  DS, AX
            MOV  BX, 150AH
            MOV  CH, 7           ；CH＝7
LOOP:       MOV  [BX], CH        ；パレット番号＝CH
            MOV  [BX+1], CH      ；カラーコード＝CH
            INT  0A4H            ；GCOLOR2
            DEC  CH              ；CH＝CH－1
            JGE  LOOP            ；IF CH>=0
            IRET
```

3．6　GCLS（描画領域の消去）

機　　能	GVIEWで設定された描画領域を消去します.カラーモードのときはバックグラウンドカラーで塗りつぶします.
絶対番地	F990H：05ACH
割　　込	INT　A5H
入力条件	DS←グラフィックLIOワークエリアのセグメントベース
引　　数	なし
出力条件	AH←0
破　　壊	DS，SS，SPを除くすべて
解　　説	

BASICのCLS　2に対応します．BASICと同様，ビューポートが設定されている場合は，ビューポート内のみ消去されます．

処　　理

描画領域が画面全体か否かで処理が異なります．画面全体の場合は，一括して0またはFFFFHを書き込みだけで済みますから，非常に高速に処理されます．GVIEWで描画領域を画面の一部に設定している場合は，領域判定のため処理速度は遅くります．

［ビューポートを設定している場合］

(1)　矩形領域の塗りつぶしルーチンをコールする．（CALL　349EH）

［ビューポートを設定していない場合］

(1)　0またはFFFFHをワード単位で書き込む．（REP：STOSW

(2)　カラーの場合は，(1)の処理中，グラフィック画面の表示を止める．

サンプル

BASICのステートメント　CLS　2を機械語で実現します．

```
MOV  AX, 0060H
MOV  DS, AX
INT  0A5H
```

3．7 GPSET（ドットのセット・リセット）

機　能	描画領域内の任意のドットのセット・リセットを行う．
絶対番地	F990H：06D9H
割　込	INT　A6H
入力条件	DS←グラフィックLIOワークエリアのセグメントベース BX←引数リストの先頭オフセットアドレス(データセグメント内) AH←1（PSET）または　2（PRESET）
引　数	[BX＋0～1]←X座標 [BX＋2～3]←Y座標 [BX＋4]←パレット番号　（省略するときは　FFH）

DS：BX+0　　　+2　　　+4　　+5

X座標	Y座標	パレット番号

出力条件	AH←0
破　壊	DS，SS，SPを除くすべて

解　説

BASICのPSET，PRESETに対応します．AHが1のとき，PSET，AHが2のとき，PRESETとなります．ドットのセット・リセットは，GVIEWで指定した領域内でのみ行なわれます．パレット番号を省略(＝FFH)すると，PSETでは，フォアグラウンドカラーが，PRESETでは，バックグラウンドカラーが使われます．

処　理

PSETルーチンをコールする．(CALL　1F5DH)

サンプル

スクリーン座標（100，50）に点を打つ．

[BASICステートメント]

```
PSET （100，50）
```

[機械語プログラム]

```
MOV AX, 0060H
MOV DS, AX
MOV BX, 150AH
MOV WORD [BX], 64H        ;X=100
MOV WORD [BX+2], 32H      ;Y=50
MOV BYTE [BX+4], 0FFH     ;省略
MOV AH, 1                 ;PSET
INT 0A6H
```

3．8 GLINE
(直線，ボックスの描画，塗りつぶし)

機能	直線，ボックスの描画，塗りつぶし等を行なう．
絶対番地	F990H：0701H
割込	INT A7H
入力条件	DS←グラフィックLIOワークエリアのセグメントベース BX←引数リストの先頭オフセットアドレス(データセグメント内)
引数	[BX+0〜1]←始点のX座標　(X1) [BX+2〜3]←始点のY座標　(Y1) [BX+4〜5]←終点のX座標　(X2) [BX+6〜7]←終点のY座標　(Y2) [BX+8]←パレット番号1　直線及びボックスの描前に使う色を指定する．(省略するときは　FFH) [BX+9]←描画コード 0：直線の描画 1：ボックスの描画(B) 2：ボックスの塗りつぶし(BF) 《描画コードが0または1の場合》 [BX+0AH]←ラインスタイル・スイッチ 0：ラインスタイル指定なし 1：ラインスタイル指定あり [BX+0BH〜0CH]←ラインスタイル

《描画コードが2（ＢＦ）の場合》

［Ｂ·Ｘ＋０ＡＨ］←パレット，タイルスイッチ

０：パレット指定なし，タイル指定なし

１：パレット指定あり（ＰＣ－９８０１では指定できない）

２：タイル指定あり（ＰＣ－９８０１では指定できない）

以下のパラメータは，ＰＣ－９８０１では指定できない．

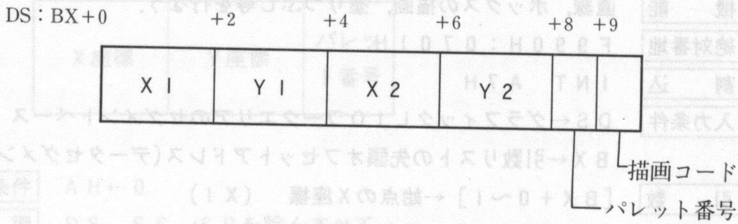
［ＢＸ＋０ＢＨ］←パレット番号２塗りつぶしに使う色を指定する．

（省略するときは　ＦＦＨ）

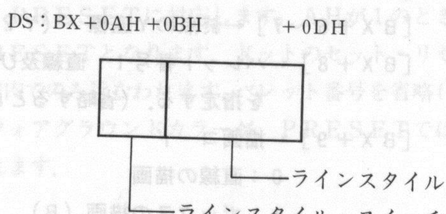
［ＢＸ＋０ＤＨ］←タイルストリングの文字数

［ＢＸ＋０ＥＨ～０ＦＨ］←タイルストリングのオフセットアドレス

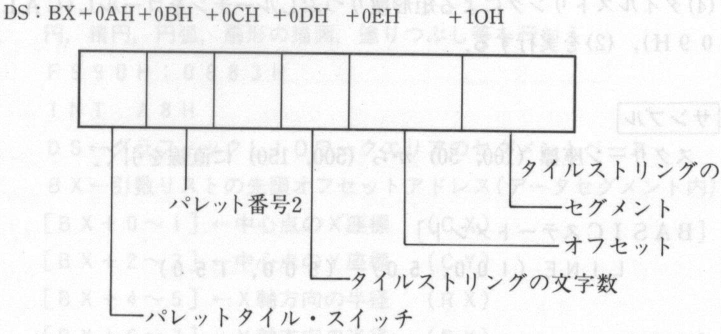
［ＢＸ＋１０Ｈ～１１Ｈ］←タイルストリングのセグメントアドレス



《描画コードが０または１の場合》



描画コードが２の場合》



出力条件　ＡＨ←０　（正常終了）

ＡＨ←５　（ＢＦ及びタイル指定がある場合で，タイルストリング の文字数が，カラーで３未満，白黒で０のとき）

破　壊　ＤＳ，ＳＳ，ＳＰを除くすべて

解　説

ＢＡＳＩＣのＬＩＮＥステートメントに対応しています．ＰＣ−９８０１Ｅ／Ｆ／Ｍでは，機能が強化され，タイルストレングによるボックスの塗りつぶし，外枠と異なる色による塗りつぶしが可能となっています．これらの機能はＰＣ−９８０１では使えませんから注意して下さい．

Ｘ座標，Ｙ座標は−３２７６８〜３２７６７の任意の点を指定できますが，実際の描画はビューポート内のみに行なわれます．

処　理

《描画コードが０の場合》

(1)　直線描画サブルーチンをコールする．（ＣＡＬＬ　２０１６Ｈ）

《描画コードが１の場合》

(2)　直線描画サブルーチンを４回コールして，ボックスを描く．

（ＣＡＬＬ　２０１６Ｈ）

《描画コードが２の場合》

〈タイルストリング指定がない場合〉

(3)矩形塗りつぶしルーチンをコールし（ＣＡＬＬ　３４９ＥＨ），(2)を実行する．

〈タイルストリング指定がある場合〉

(4)タイルストリングによる矩形塗りつぶしルーチンをコールし(CALL 2609H)，(2)を実行する．

サンプル

スクリーン座標（100，50）から（500，150）に直線を引く．

[BASICステートメント]

```
LINE(100,50)-(500,150)
```

[機械語によるプログラム]

```
MOV AX, 0060H
MOV DS, AX
MOV BX, 150AH
MOV WORD [BX], 64H      ;X1=100
MOV WORD [BX+2], 32H    ;Y1=50
MOV WORD [BX+4], 1F4H   ;X2=500
MOV WORD [BX+6], 96H    ;Y2=150
MOV BYTE [BX+8], 0FFH   ;色=省略
MOV BYTE [BX+9], 0      ;直線
MOV BYTE [BX+0AH],      ;ラインスタイルなし
INT 0A7H
```

3．9　GCIRCLE（円，円弧の描画）

機　能	円，楕円，円弧，扇形の描画，塗りつぶし等を行なう．
絶対番地	F990H：0883H
割　込	INT　A8H
入力条件	DS←グラフィックLIOワークエリアのセグメントベース BX←引数リストの先頭オフセットアドレス(データセグメント内)
引　数	[BX＋0～1]←中心点のX座標　(CX)

[BX＋2～3]←中心点のY座標　(CY)

[BX＋4～5]←X軸方向の半径　(RX)

[BX＋6～7]←Y軸方向の半径　(RY)

[BX＋8]←パレット番号1〈線の色〉(省略するときはFFH)

[BX＋9]←描画コード（各ビットが次の意味を持つ）

ビット0：開始点指定　0→なし，1→あり

ビット1：開始点半径描画　0→なし，1→あり

ビット2：終了点指定　0→なし，1→あり

ビット3：終了点半径描画　0→なし，1→あり

ビット4：開始点・終了点一致の描画

0→円，1→一致点のみ

ビット5：塗りつぶし指定　0→なし，1→あり

（PC－9801では指定できない）

ビット6：タイル指定　0→なし，1→あり

（PC－9801では指定できない）

[BX＋0AH～0BH]←開始点のX座標　(SX)

[BX＋0CH～0DH]←開始点のY座標　(SY)

[BX＋0EH～0FH]←終了点のX座標　(EX)

[BX＋10H～11H]←終了点のY座標　(EY)

《塗りつぶし指定あり，タイル指定なしのとき》

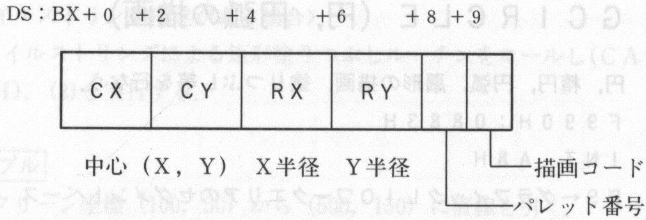
[BX＋12H]←パレット番号2〈領域色〉(省略はFFH)

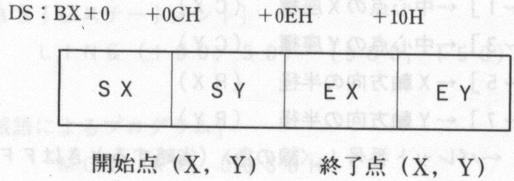
《塗りつぶし指定あり，タイル指定ありのとき》

[BX＋12H]←タイルストリングの文字数

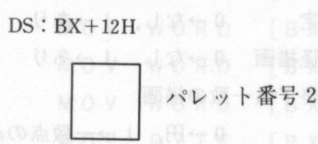
[BX＋13H～14H]←タイルストリングのオフセットアドレス

[BX＋15H～16H]←タイルストリングのセグメントアトレス

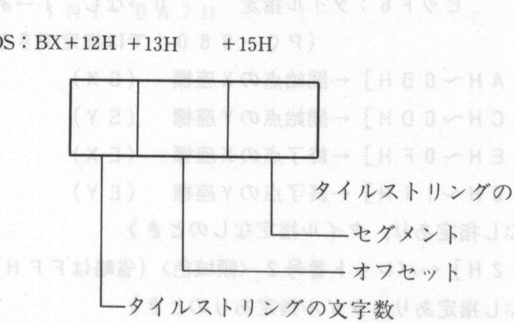




《塗りつぶし指定あり，タイル指定なしのとき》



《塗りつぶし指定あり，タイル指定ありのとき》



出力条件　ＡＨ←０　（正常終了）
ＡＨ←５　（塗りつぶし及びタイル指定がある場合で，タイルストリングの文字数が，カラーで３未満，白黒で０のとき）

AH←6　（座標計算の結果，座標が整数型2バイトで表せない値となったとき）

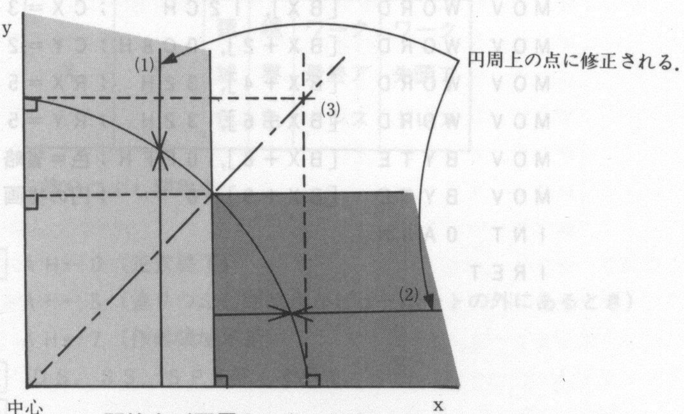
破　壊　DS，SS，SPを除くすべて

解　説

BASICのCIRCLEステートメントに対応していますが，引数の意味が少し異なります．楕円の場合，BASICでは比率で表すのに対し，LIOでは直接，X半径，Y半径で表します．また，円弧，扇形の場合，開始点，終了点をBASICでは角度で表わすのに対し，LIOでは円周上の座標で表わします．

PC9801E／F／Mでは，機能が強化され，塗りつぶしも可能となっています．

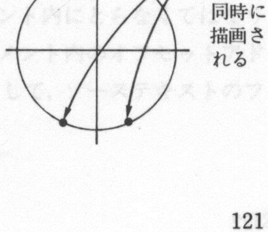
開始点，終了点は原則として円周上の点であることが必要ですが，万一，円周上以外の点を指定した場合の扱いは次のようになります．図で開始点または終了点が(1)の範囲にあるときは，Y座標が修正されて円周上の点が仮定されます．(2)の範囲にあるときは，逆にX座標が修正されて円周上の点が仮定されます．(3)の範囲にあるときは満足な描画が行なわれません．



開始点が円周上にないとき

処　理

本ルーチンによる円の描画にはGDCの機能は使われておらず，プログラム中で各点の座標を計算し，直接GVRAMに書き込んでいます．



(1)　円周上の各点の座標を計算する．

(2)　対称な4点を同時に描画する．

(3) 各点の描画は点を打つルーチン（1F5DH）をコールする.

(4) 塗りつぶし指定がある場合は2609HをコールしてX軸に平行な線を引く.

(5) 開始点，終了点線分の指定がある場合は，円弧の描画後，線を引くルーチン（2016H）をコールする.

注　意

本ルーチンには，INT　C5Hが埋め込まれていますので，対応する処理ルーチンが準備されていることが必要です.（3．0参照）N88－BASICでは，これをSTOPキーのチェックに使っており，描画中にSTOPキーの押下があるとBASICのコマンドの待ち状態に戻ってしまいます.

サンプル

中心（300，200），半径50の円を描きます.

```
MOV  AX, 0060H
MOV  DS, AX
MOV  BX, 150AH
MOV  WORD  [BX], 12CH     ;CX=300
MOV  WORD  [BX+2], 0C8H;CY=200
MOV  WORD  [BX+4], 32H   ;RX=50
MOV  WORD  [BX+6], 32H   ;RY=50
MOV  BYTE  [BX+8], 0FFH;色=省略
MOV  BYTE  [BX+9], 0     ;円の描画
INT  0A8H
IRET
```

3．10　GPAINT1（指定色による塗りつぶし）

機　能　指定された境界色で囲まれた領域を，指定色で塗りつぶす．

絶対番地　F990H：12D0H

割　込　INT　A9H

入力条件　DS←グラフィックLIOワークエリアのセグメントベース

BX←引数リストの先頭オフセットアドレス（データセグメント内）

引　数　［BX＋0～1］←塗りつぶし開始点のX座標

［BX＋2～3］←塗りつぶし開始点のY座標

［BX＋4］←パレット番号〈領域色〉　（省略はFFH）

［BX＋5］←パレット番号〈境界色〉　（省略はFFH）

［BX＋6～7］←作業領域最終オフセットアドレス

［BX＋8～9］←作業領域先頭オフセットアドレス

DS：BX+0	+2	+4	+5	+6	+8
X	Y	領域色	境界色	ワーク最終アドレス	ワーク先頭アドレス

塗りつぶし開始点

出力条件　AH←0（正常終了）

AH←5（塗りつぶし開始点がビューポートの外にあるとき）

AH←7（作業領域不足）

破　壊　DS，SS，SPを除くすべて

解　説

BASICの〔1〕PAINTに対応しています．〈境界色〉で囲まれた領域を〈領域色〉で塗りつぶします．〈領域色〉が省略された場合は，〈領域色〉と同じとみなされます．

作業領域は16バイト以上必要で，データセグメント内にとらなくてはなりません．その先頭および最終アドレスを，データセグメント内のオフセットアドレスで指定します．N88－BASICでは，作業領域として，ソーステキストのフリーエリアを指定しています．

注　意

本ルーチンには，ＩＮＴ　Ｃ５Ｈが埋め込まれていますので，対応する処理ルーチンが準備されていることが必要です．（３．０参照）N88－ＢＡＳＩＣではこれをＳＴＯＰキーのチェックに使っており，描画中にＳＴＯＰキーの押下があるとＢＡＳＩＣのコマンド待ちの状態に戻ってしまいます．

処　理

塗りつぶしの内部サブルーチンをコールします．（ＣＡＬＬ　２１Ｆ０Ｈ）

サンプル

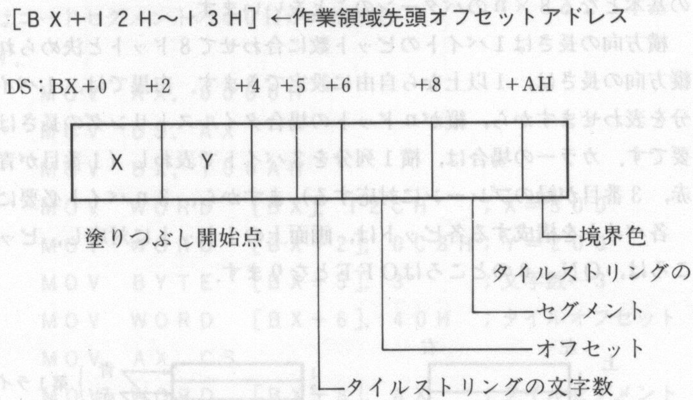
前節のサンプルで描いた円を緑色で塗りつぶします．作業領域は，ＢＡＳＩＣと同じようにテキストの空きエリアを使います．空きエリアの先頭は，セグメント６０Ｈの［６Ａ６Ｈ～６Ａ７Ｈ］に，空きエリアの最終は，［６Ａ８Ｈ～６Ａ９Ｈ］に格納されています．

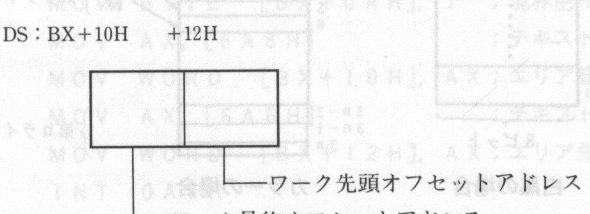
```
MOV  AX, 0060H
MOV  DS, AX
MOV  BX, 150AH
MOV  WORD  [BX], 12CH       ; X=300
MOV  WORD  [BX+2], 0C8H     ; Y=200
MOV  BYTE  [BX+4], 4        ; 領域色←緑
MOV  BYTE  [BX+5], 7        ; 境界色←白
MOV  AX, [6A8H]             ; テキストフリー
MOV  WORD  [BX+6], AX       ; エリア最終
MOV  AX  [6A6H]             ; テキストフリー
MOV  WORD  [BX+8], AX       ; エリア先頭
INT  0A9H
```

3. 11 GPAINT2

(タイルパターンによる塗りつぶし)

機　能	指定された境界色で囲まれた領域を，タイルパターンで塗りつぶす．
絶対番地	F990H：13AAH
割　込	INT　AAH
入力条件	DS←グラフィックLIOワークエリアのセグメントベース BX←引数リストの先頭オフセットアドレス(データセグメント内)
引　数	［BX＋0～1］←塗りつぶし開始点のX座標 ［BX＋2～3］←塗りつぶし開始点のY座標 ［BX＋5H］←タイルストリングの文字数 ［BX＋6H～7H］←タイルストリングのオフセットアドレス ［BX＋8H～9H］←タイルストリングのセグメントアドレス ［BX＋0AH］←パレット番号〈境界色〉(省略はFFH) ［BX＋10H～11H］←作業領域最終オフセットアドレス ［BX＋12H～13H］←作業領域先頭オフセットアドレス





出力条件	ＡＨ←０	（正常終了）
	ＡＨ←５	（塗りつぶし開始点がビューポートの外にあるとき，または，タイルストリングの文字数が，カラーで３未満，白黒で０のとき）
	ＡＨ←７	（作業領域不足）
破　壊	ＤＳ，ＳＳ，ＳＰを除くすべて	

解　説

ＢＡＳＩＣの〔２〕ＰＡＩＮＴに対応しています．〈境界色〉を省略するとフォアグラウンドカラーが使われます．

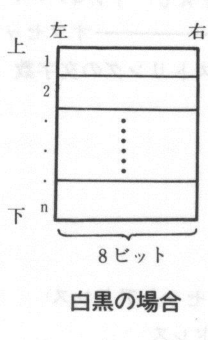
作業領域は１６バイト以上必要で，データセグメント内にとらなくてはなりません．その先頭および最終アドレスを，データセグメント内のオフセットアドレスで指定します．N88－ＢＡＳＩＣでは，作業領域として，ソーステキストのフリーエリアを指定しています．

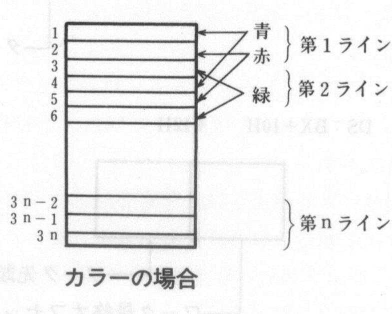
タイトルストリングの構成

タイトルストリングによる塗りつぶしは，横８ドット，縦ｎドットの長方形のパターンを繰り返し使って指定した領域を埋めます．タイルストリングとは，その基本となる８×ｎのパターンのことをいいます．

横方向の長さは１バイトのビット数に合わせて８ドットと決められていますが，縦方向の長さは，１以上なら自由に設定できます．白黒では，１バイトで横１列分を表わせますから，縦がｎドットの場合タイルストリングの長さはｎバイト必要です．カラーの場合は，横１列分を３バイトで表わし（１番目が青，２番目が赤，３番目が緑のプレーンに対応する）ますから，３ｎバイト必要になります．

各バイトを構成する各ビットは，画面上の１ドットに対応し，ビットが１のところは，ＯＮ，０のところはＯＦＦとなります．





注　意

本ルーチンには，ＩＮＴ　Ｃ５Ｈが埋め込まれていますので，対応する処理ルーチンが準備されていることが必要です．（３．０参照）N88－ＢＡＳＩＣでは，これをＳＴＯＰキーのチェックに使っており，描画中にＳＴＯＰキーの押下があるとＢＡＳＩＣのコマンド待ちの状態に戻ってしまいます．

処　理

塗りつぶしの内部サブルーチンをコールします．（ＣＡＬＬ　２１Ｆ０Ｈ）

サンプル

３．９のサンプルで描いた円を右のようなパターンで塗りつぶします．作業領域は，ＢＡＳＩＣと同じようにテキストの空きエリアを使います．空きエリアの先頭は，セグメント６０Ｈの［６Ａ６Ｈ～６Ａ７Ｈ］に，空きエリアの最終は，［６Ａ８Ｈ～６Ａ９Ｈ］に格納されています．タイルストリングはコードセグメントの４０Ｈから格納しておきます．

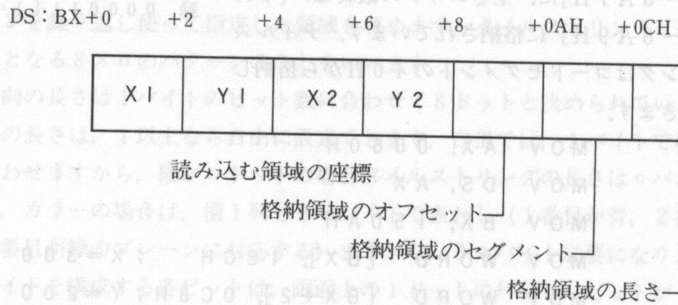
黒青赤紫緑水色黄白

青　０１０１０１０１＝５５Ｈ
赤　００１１００１１＝３３Ｈ
緑　００００１１１１＝０ＦＨ

```
MOV  AX, 0060H
MOV  DS, AX
MOV  BX, 150AH
MOV  WORD  [BX], 12CH     ;X=300
MOV  WORD  [BX+2], 0C8H;Y=200
MOV  BYTE  [BX+5], 3      ;文字数←3
MOV  WORD  [BX+6], 40H    ;タイルオフセット
MOV  AX, CS
MOV  WORD  [BX+8], AX     ;タイルセグメント
MOV  BYTE  [BX+0AH], 7    ;境界色←7
MOV  AX, [6A8H]           ;テキストフリー
MOV  WORD  [BX+10H], AX;エリア最終
MOV  AX, [6A6H]           ;テキストフリー
MOV  WORD  [BX+12H], AX;エリア先頭
INT  0A9H
 .
ORG  40H
DB   55H, 33H, 0FH
```

3．12　GGET（グラフィックの読み込み）

機　能	画面上のグラフィックパタンをメモリに読み込みます．
絶対番地	F990H：1466H
割　込	INT　ABH
入力条件	DS←グラフィックLIOワークエリアのセグメントベース BX←引数リストの先頭オフセットアドレス（データセグメント内）
引　数	［BX＋0～1］←読み込む領域の左上のX座標（X1） ［BX＋2～3］←読み込む領域の左上のY座標（Y1） ［BX＋4～5］←読み込む領域の右下のX座標（X2） ［BX＋6～7］←読み込む領域の右下のY座標（Y2） ［BX＋8～9］←格納領域のオフセットアドレス ［BX＋0AH～0BH］←格納領域のセグメントアドレス ［BX＋0CH～0DH］←格納領域の長さ（バイト）



出力条件	AH←0　（正常終了） AH←5　（読み込む領域がビューポートをはみ出している場合，または，格納領域の長さが足りない場合．）
破　壊	DS，SS，SPを除くすべて

解　説

BASICのGET@ステートメントに対応しています．格納領域の構成を以下に示します．

格納領域

格納領域の先頭には，まず，読み込み領域の大きさがX，Y方向各2バイトで格納されます．そして，格納領域の4バイト目から実際のデータが格納されます．

読み込み領域の上の方から左端がバイトの先頭になるようにシフトさせながら1列ずつ格納されます．横1列のドット数が8の倍数でないときは各列の最後のバイトの余ったビットは不定となります．

《白黒の場合》

［0～1］　X方向ドット数（＝X2－X1＋1）

［2～3］　Y方向ドット数（＝Y2－Y1＋1）

以下，横1列単位でグラフィックパタンを格納

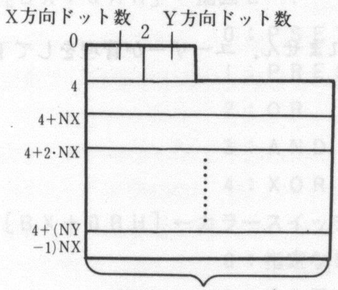
横1列のバイト数＝（X方向ドット数＋7）／8

Y方向のドット数分繰り返す．

必要とする格納領域の長さは，

4＋横1列のバイト数＊Y方向のドット数

となる．



横1列のバイト数＝NX

ただしNYはY方向ドット数

白黒の場合

《カラーの場合》

［0～1］　X方向ドット数（＝X2－X1＋1）

［2～3］　Y方向ドット数（＝Y2－Y1＋1）

以下，横1列単位で青，赤，緑の順にグラフィックパタンを格納

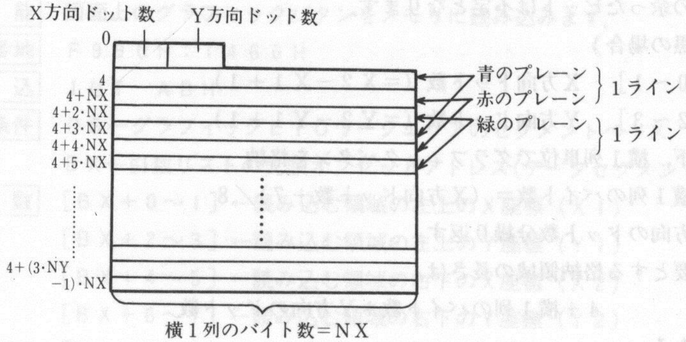
横1列のバイト数＝（X方向ドット数＋7）／8＊3

Y方向のドット数分繰り返す．

必要とする格納領域の長さは，

4＋横1列のバイト数＊Y方向のドット数

となる．



カラーの場合

画面モードは，格納領域にストアされません．ユーザーが管理をして下さい．

3．13　GPUTI（グラフィックの書き込み）

機　能　メモリに格納されているグラフィックパタンを画面に書き込みます.

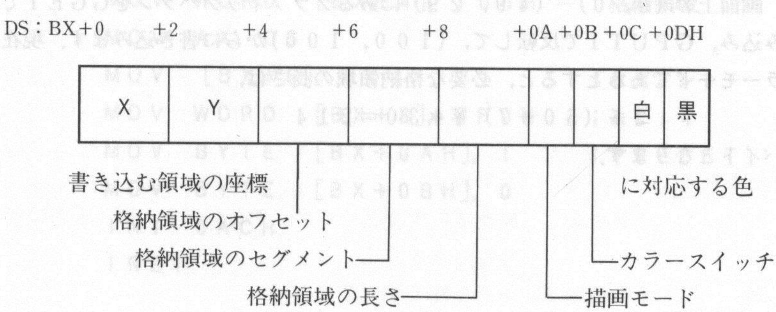
絶対番地　F990H：1605H

割　込　INT ACH

入力条件　DS←グラフィックLIOワークエリアのセグメントベース
BX←引数リストの先頭オフセットアドレス(データセグメント内)

引　数　[BX＋0～1] ←書き込む領域の左上のX座標
[BX＋2～3] ←書き込む領域の左上のY座標
[BX＋4～5] ←格納領域のオフセットアドレス
[BX＋6～7] ←格納領域のセグメントアドレス
[BX＋8～9] ←格納領域の長さ（バイト）
[BX＋0AH] ←描画モード
0：PSET
1：PRESET
2：OR
3：AND
4：XOR
[BX＋0BH] ←カラースイッチ
0：指定なし
1：白と黒に色をつける

《カラースイッチが1のとき以下のパラメータが有効となる》
[BX＋0CH] ←白の部分につける色（パレット番号）
[BX＋0DH] ←黒の部分につける色（パレット番号）



出力条件　ＡＨ←０　（正常終了）

ＡＨ←５　（書き込むグラフィックパターンがビューポート内に収まらないとき）

破　壊　ＤＳ，ＳＳ，ＳＰを除くすべて

解　説

ＢＡＳＩＣのＰＵＴ＠ステートメント（配列）に対応しています．ＧＧＥＴでメモリに格納したグラフィックパタンを画面に戻すのに使います．

描画モードを０とすると，格納データはそのまま書き込まれ，１にすると反転して書き込まれます．２～４は，画面上のデータと格納データとの演算結果が書き込まれていないので，格納時の画面モードにかかわらず現在のモードで再現されます．ですから，カラーで格納したデータを白黒で再現，または，その逆を行なうと正常に作動しません．白黒のデータをカラーで再現する場合に限っては，カラースイッチを１とすることにより対処できます．

カラースイッチは白黒モードで格納したデータをカラーモードで再現する場合に１にします．そのとき，元のデータで格納したデータの白の部分につける色を［ＢＸ＋０ＣＨ］で，黒の部分につける色を［ＢＸ＋０ＤＨ］で指定します．

注　意

本ルーチンには，ＩＮＴ　Ｃ５Ｈが埋め込まれていますので，対応する処理ルーチンが準備されていることが必要です．（３．０参照）　N_{88}－ＢＡＳＩＣでは，これをＳＴＯＰキーのチェックに使っており，描画中にＳＴＯＰキーの押下があるとＢＡＳＩＣのコマンド待ちの状態に戻ってしまいます．

処　理

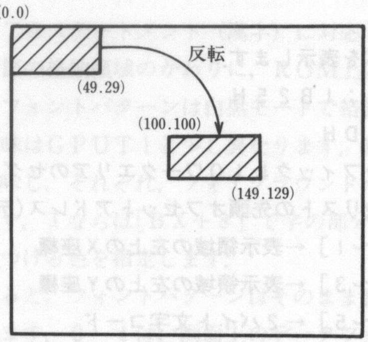
ＰＵＴの内部サブルーチンをコールします．（ＣＡＬＬ　２６６９Ｈ）

サンプル

画面上の（０，０）－（４９，２９）にあるグラフィックパタンをＧＧＥＴで読み込み，ＧＰＵＴ１で反転して，（１００，１００）からに書き込みます．現在カラーモードであるとすると，必要な格納領域の長さは，

４＋（５０＋７）￥＊３０＝２１４

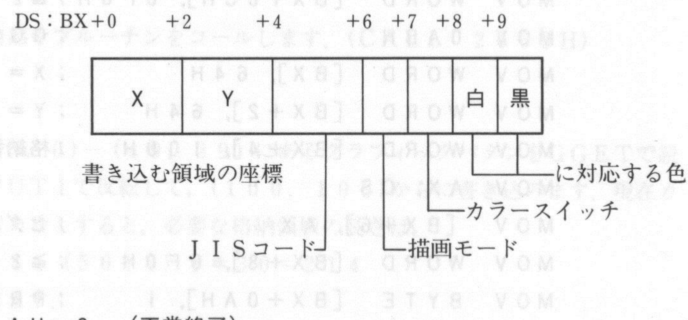
バイトとなります．




```
MOV  AX, 0060H
MOV  DS, AX
MOV  BX, 150AH
MOV  WORD  [BX], 0          ;X1
MOV  WORD  [BX+2], 0        ;Y1
MOV  WORD  [BX+4], 31H      ;X2=49
MOV  WORD  [BX+6], 1DH      ;Y2=29
MOV  WORD  [BX+8], 100H     ;格納領域
MOV  AX, CS
MOV  [BX+0AH], AX           ;セグメント
MOV  WORD  [BX+0CH], 0F0H;≧214
MOV  0ABH                   ;GGET
MOV  WORD  [BX], 64H        ;X=100
MOV  WORD  [BX+2], 64H      ;Y=100
MOV  WORD  [BX+4], 100H     ;格納領域
MOV  AX, CS
MOV  [BX+6], AX             ;セグメント
MOV  WORD  [BX+8], 0F0H     ;≧214
MOV  BYTE  [BX+0AH], 1      ;PRESET
MOV  BYTE  [BX+0BH], 0
INT  0ACH
IRET
```

3．14　GPUT2（画面に漢字を表示）

機　　能	画面に漢字を表示します．
絶対番地	F990H：1B25H
割　　込	INT　ADH
入力条件	DS←グラフィックLIOワークエリアのセグメントベース BX←引数リストの先頭オフセットアドレス(データセグメント内)
引　　数	［BX＋0～1］←表示領域の左上のX座標 ［BX＋2～3］←表示領域の左上のY座標 ［BX＋4～5］←2バイト文字コード ［BX＋6］←描画モード 0：PSET 1：PRESET 2：OR 3：AND 4：XOR ［BX＋7］←カラースイッチ 0：色の指定を省略 1：以下のパラメータで色を指定 ［BX＋8］←字の部分につける色　（パレット番号） ［BX＋9］←背景の部分につける色（パレット番号）



出力条件	AH←0　（正常終了） AH←5　（書き込むグラフィックパタンがビューポート内に収まらないとき）
破　　壊	DS，SS，SPを除くすべて

解　説

ＢＡＳＩＣのＰＵＴ＠ステートメント（漢字）に対応しています．

本ルーチンは，前節の格納領域のかわりに，ＲＯＭ上にあるフォントパターンを用いるものです．フォントパターンは白黒モードで格納されたものに相当し，カラースイッチの意味はＧＰＵＴ１と少し異なります．０ならば，［ＢＸ＋８，９］の色指定の省略を意味し，それぞれ，フォアグラウンドカラー，バックグラウンドカラーが使われます．１ならば［ＢＸ＋８］で字の部分につける色を，［ＢＸ＋９］で背景の部分につける色を指定します．

描画モード０とすると，フォントパターンはそのまま書き込まれ，１にすると反転して書き込まれます．２～４は，画面上のデータフォントパターンとの演算結果が書き込まれます．

文字のコードはＪＩＳ漢字コードだけでなく，００ＸＸの半角コード，０１ＸＸの１／４角コードも使うことができます．（次ページのコード表参照）

注　意

本ルーチンには，ＩＮＴ　Ｃ５Ｈが埋め込まれていますので，対応する処理ルーチンが準備されていることが必要です．（３．０参照）N88－ＢＡＳＩＣでは，これをＳＴＯＰキーのチェックに使っており，描画中にＳＴＯＰキーの押下があるとＢＡＳＩＣのコマンド待ちの状態に戻ってしまいます．

処　理

文字のフォントパターンを読み出した後，ＧＰＵＴ１と同様にＰＵＴの内部サブルーチンをコールします．（ＣＡＬＬ　２６６９Ｈ）

フォントパターンの読み出しは，基本的にはＢＩＯＳのフォントパターンの読み出しルーチン（ＩＮＴ　１８Ｈ，ＡＨ＝１４Ｈ）を使いますが，ＢＩＯＳで扱うコードとこのルーチンの受けるコードが一部異なるため変更が必要とります．また，ひらがなは漢字ＲＯＭに含まれていないため，別に扱います．

(1) ＪＩＳ漢字コードの場合

文字コードの変換なし

(2) 半角文字（ひらがなを除く）の場合

グラフィックでは，半角文字のコードは００ＸＸですが，ＢＩＯＳでは，アルファベットが２８ＸＸ，カタカナが２９ＸＸのコードで表現されています．

変換式は次のようになります．

〈アルファベットの場合（２０Ｈ≦ＣＯＤＥ≦７ＦＨ)〉

ＢＩＯＳＣＯＤＥ←ＣＯＤＥ＋２８００Ｈ

〈カタカナの場合（０Ａ０Ｈ≦ＣＯＤＥ≦０ＤＦＨ）〉

ＢＩＯＳＣＯＤＥ←ＣＯＤＥ＋２８８０Ｈ

(3) １／４角文字の場合

グラフィックでは，０１ＸＸですが，ＢＩＯＳでは，００ＸＸとなるので，頭の１バイトを００に置き換えます．

(4) ひらがなの場合

漢字ＲＯＭ中にはフォントパターンがなく，ＬＩＯのＲＯＭに含まれています．具体的には，本ルーチンの直前，つまり，セグメントＦ９９０Ｈの１７０５Ｈから１Ｂ２４Ｈにあります．ひらがなの場合にはＢＩＯＳをコールせず，直接この領域からフォントパターンを転送します．次ページの半角文字の表で斜線の部分がここでいうひらがなに相当します．

サンプル

画面の（１００，１００）に"漢"という字を赤色で出力します．

```
MOV  AX, 0060H
MOV  DS, AX
MOV  BX, 150AH
MOV  WORD  [BX], 64H         ; X=100
MOV  WORD  [BX+2], 64H       ; Y=100
MOV  WORD  [BX+4], 3441H     ; 漢のコード
MOV  BYTE  [BX+6], 0         ; PSET
MOV  BYTE  [BX+7], 1         ; 色指定あり
MOV  BYTE  [BX+8], 2         ; 字の部分←赤
MOV  BYTE  [BX+9], 0         ; 背景←黒
INT  0ADH
         ・
         ・
```

グラフィック画面に表示できる半角文字一覧表

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0020	Ⓢⓟ	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0030	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0040	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0050	P	Q	R	S	T	U	V	W	X	Y	Z	[	¥	]	^	_
0060		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0070	p	q	r	s	t	u	v	w	x	y	z	{	¦	}	~	
0080		。	「	」	、	・	を	ぁ	ぃ	ぅ	ぇ	ぉ	ゃ	ゅ	ょ	っ
0090	ー	あ	い	う	え	お	か	き	く	け	こ	さ	し	す	せ	そ
00A0	Ⓢⓟ	。	「	」		・	ヲ	ァ	ィ	ゥ	ェ	ォ	ャ	ュ	ョ	ッ
00B0	ー	ア	イ	ウ	エ	オ	カ	キ	ク	ケ	コ	サ	シ	ス	セ	ソ
00C0	タ	チ	ツ	テ	ト	ナ	ニ	ヌ	ネ	ノ	ハ	ヒ	フ	ヘ	ホ	マ
00D0	ミ	ム	メ	モ	ヤ	ユ	ヨ	ラ	リ	ル	レ	ロ	ワ	ン	゛	゜
00E0	た	ち	つ	て	と	な	に	ぬ	ね	の	は	ひ	ふ	へ	ほ	ま
00F0	み	む	め	も	や	ゆ	よ	ら	り	る	れ	ろ	わ	ん	゛	゜
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

▒の部分は漢字ROMにない。メインROMに
フォントパターンが格納されている。

グラフィック画面に表示できる1/4角文字一覧表

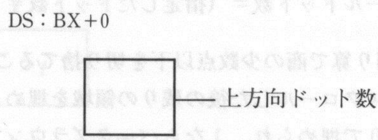
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0100		S_H	S_X	E_X	E_T	E_Q	A_K	B_L	B_S	H_T	L_F	H_M	C_L	C_R	S_O	S_I
0110	D_E	D_1	D_2	D_3	D_4	N_K	S_N	E_B	C_N	E_M	S_B	E_C	→	←	↑	↓
0120	(SP)	!	"	#	$	%	&	'	(	)	*	+	,	−	.	/
0130	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0140	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0150	P	Q	R	S	T	U	V	W	X	Y	Z	[	¥	]	∧	—
0160		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0170	p	q	r	s	t	u	v	w	x	y	z	{	¦	}	~	DEL
0180																＋
0190													╭	╮	╰	╯
01A0	(SP)	。	「	」	、	・	ヲ	ァ	ィ	ゥ	ェ	ォ	ャ	ュ	ョ	ッ
01B0	ー	ア	イ	ウ	エ	オ	カ	キ	ク	ケ	コ	サ	シ	ス	セ	ソ
01C0	タ	チ	ツ	テ	ト	ナ	ニ	ヌ	ネ	ノ	ハ	ヒ	フ	ヘ	ホ	マ
01D0	ミ	ム	メ	モ	ヤ	ユ	ヨ	ラ	リ	ル	レ	ロ	ワ	ン	゛	゜
01E0	═	╞	╪	╡					♠	♥	♦	♣	●	○	／	＼
01F0	×	円	年	月	日	時	分	秒								DEL
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

3．15　GROLL（グラフィック画面のスクロール）

機　能	グラフィック画面をスクロールさせる.
絶対番地	F990H：1C65H
割　込	INT　AEH
入力条件	DS←グラフィックLIOワークエリアのセグメントベース BX←引数リストの先頭オフセットアドレス(データセグメント内)
引　数	

《PC9801の場合》

［BX＋0～1］←上方向ドット数



《PC9801E／Fの場合》

［BX＋0～1］←上下方向ドット数

正：上にスクロール

負：下にスクロール

［BX＋2～3］←左右方向ドット数

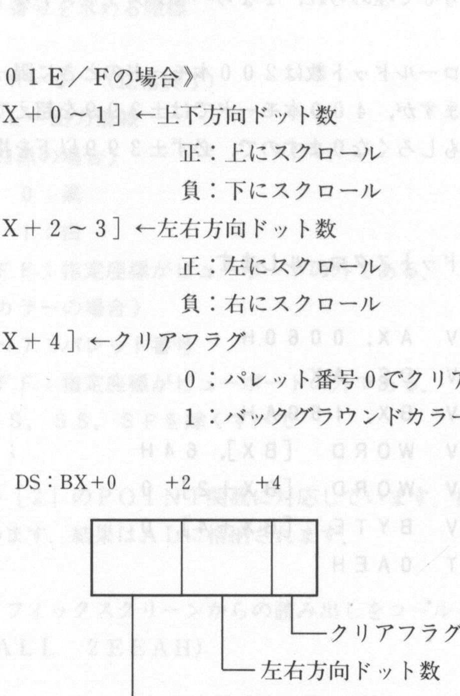
正：左にスクロール

負：右にスクロール

［BX＋4］←クリアフラグ

0：パレット番号0でクリア

1：バックグラウンドカラーでクリア



出力条件　ＡＨ←０　（正常終了）
ＡＨ←５（２００本モードにおいて，上下方向のドット数が±１９９を超えているとき）

破　壊　ＤＳ，ＳＳ，ＳＰを除くすべて

解　説

ＢＡＳＩＣのＲＯＬＬステートメントに対応しています．ＰＣ９８０１では，上方向へのスクロールしかできませんでしたが，ＰＣ９８０１Ｅ／Ｆ／Ｍでは，大幅に機能が拡張されて，上下方向および左右方向にもスクロールできるようになっています．ただし，左右方向のスクロールはバイト単位で行われるので，ドット単位で指定しても実際にスクロールされるのは，８の倍数のドット数となります．つまり，

実際のスクロールドット数＝（指定したドット数￥８）＊８

となります．（￥は割り算で商の少数点以下を切り捨てることを意味する）

クリアフラグは，スクロールした後の残りの領域を埋める色を指定するもので，０ならパレット番号０で埋められ，１ならバックグラウンドカラーで埋められます．

上下方向のスクロールドット数は２００本モードのときに限って±１９９を超えると異常終了しますが，４００本モードでは±３９９を超えても作動します．しかし，画面がおもしろくなりますので，必ず±３９９以下を指定して下さい．

サンプル

上方向に１００ドットスクロールします．

```
MOV  AX, 0060H
MOV  DS, AX
MOV  BX, 150AH
MOV  WORD  [BX], 64H          ;100
MOV  WORD  [BX+2], 0
MOV  BYTE  [BX+4], 0
INT  0AEH
   .
   .
```

３．１６　ＧＰＯＩＮＴ

（指定座標のドットの色を得る）

機　能　指定された座標に表示されているドットの色を与えます．

絶対番地　Ｆ９９０Ｈ：１Ｅ１０Ｈ

割　込　ＩＮＴ　ＡＦＨ

入力条件　ＤＳ←グラフィックＬＩＯワークエリアのセグメントベース

ＢＸ←引数リストの先頭オフセットアドレス（データセグメント内）

引　数　［ＢＸ＋０～１］←Ｘ座標

［ＢＸ＋２～３］←Ｙ座標

DS：BX+0　+2

+0	+2
X	Y

パレット番号を求める座標

出力条件　ＡＨ←０　（正常終了）

ＡＬ←出力情報

《白黒の場合》

０：黒

１：白

ＦＦ：指定座標がビューポートの外である．

《カラーの場合》

０～７：パレット番号

ＦＦ：指定座標がビューポートの外である．

破　壊　ＤＳ，ＳＳ，ＳＰを除くすべて

解　説

ＢＡＳＩＣの［２］のＰＯＩＮＴ関数に対応しています．指定した座標のパレット番号を求めます．結果はＡＬに格納されます．

処　理

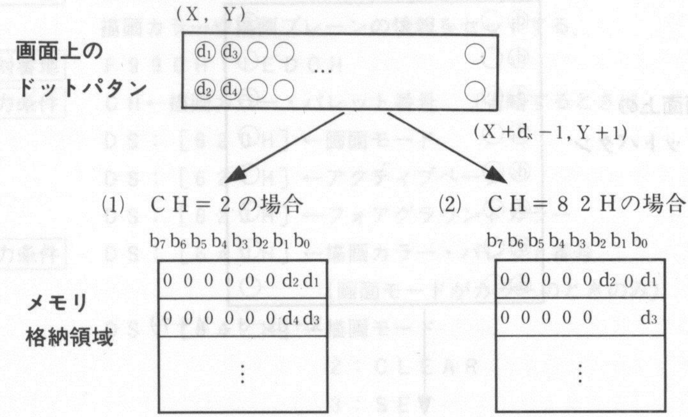
ＢＩＯＳグラフィックスクリーンからの読み出しをコールし，１ドットを読み込みます．（ＣＡＬＬ　２ＥＥＡＨ）

3．17 GCOPY
（画面上の指定領域をメモリに格納）

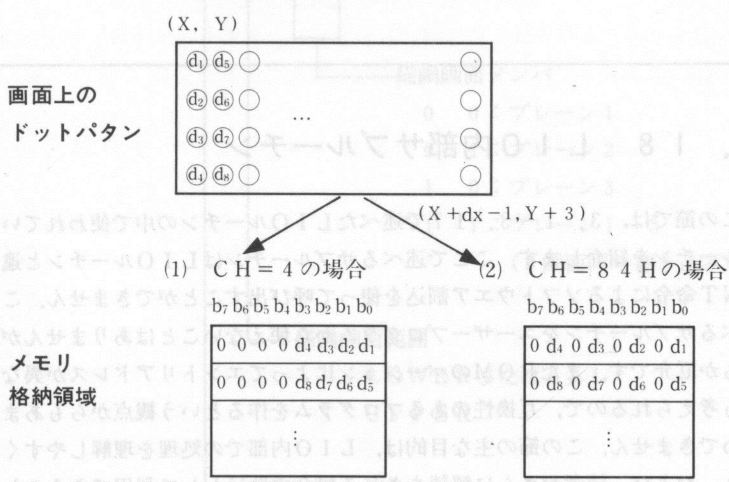
機　　能	画面上の指定領域を横を縦に変換しながら，メモリに格納する．
絶対番地	F990H：37CAH
割　　込	INT CEH
入力条件	AX←指定領域の左上のX座標　（8の倍数であること） BX←指定領域の左上のY座標 CL←指定領域のX方向ドット数（8の倍数であること） CH←指定領域のY方向ドット数 （2，4，8，82H，84Hに限る） DI←格納領域のオフセットアドレス ES←格納領域のセグメントアドレス
出力条件	AH←0　（正常終了）
破　　壊	DS，SS，SPを除くすべて
解　　説	

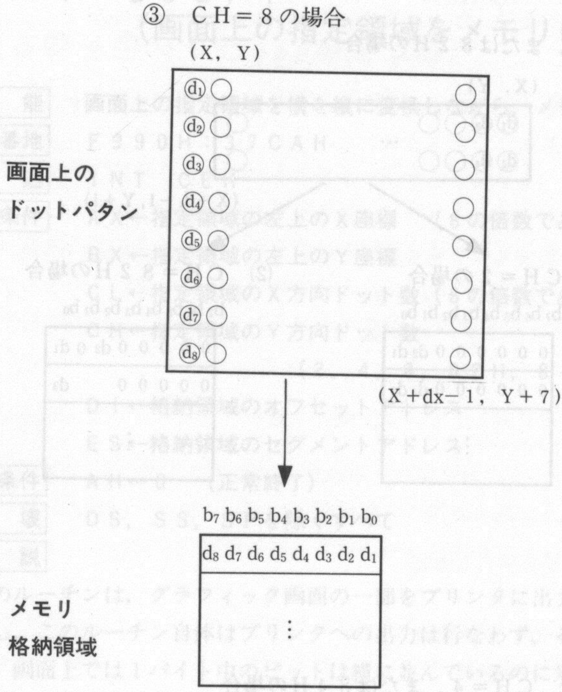
このルーチンは，グラフィック画面の一部をプリンタに出力するのに使います．ただし，このルーチン自体はプリンタへの出力は行なわず，その前処理を担当します．画面上では1バイト中のビットは横に並んでいるのに対して，プリンタへ送るグラフィックイメージは縦の8ドットを1単位としているため，横から縦への変換が必要となるわけです．このルーチンではその変換を行っているのです．一度に変換できる量はY方向ドット数が2，4，8ドットのいずれか，X方向ドット数が8の倍数で255以下です．Y方向ドット数が2，4ドットの場合はCHのb_7を1にすると1ドットおきとなります．

① ＣＨ＝２，または８２Ｈの場合



② ＣＨ＝４，または８４Ｈの場合





3．18　LIO内部サブルーチン

この節では，3．1～3．17で述べたLIOルーチンの中で使われているサブルーチンを紹介します．ここで述べるサブルーチンはLIOルーチンと違って，INT命令によるソフトウエア割込を使って呼び出すことができません．ここで述べるサブルーチンをユーザープログラムから使えないことはありませんが，いくらか厄介です．またROMのバージョンによってエントリアドレスが異なることも考えられるので，互換性のあるプログラムを作るという観点からもあまりお勧めできません．この節の主な目的は，LIO内部での処理を理解しやすくすること，および，読者がさらに解読をされる場合の助けとして利用できることにあります．

＊＊＊＊＊描画の準備＊＊＊＊（ＩＥＤＣＨ）

機　能　描画関係の内部サブルーチンのコールに先だってワークエリアに描画カラーや描画プレーンの情報をセットする．

絶対番地　Ｆ９９０Ｈ：ＩＥＤＣＨ

入力条件　ＣＨ←描画カラー・パレット番号　（省略するときは，ＦＦＨ）

ＤＳ：［６２０Ｈ］←画面モード

ＤＳ：［６２１Ｈ］←アクティブページ

ＤＳ：［６２３Ｈ］←フォアグラウンドカラー

出力条件　ＤＳ：［６４０Ｈ］←描画カラー・パレット番号

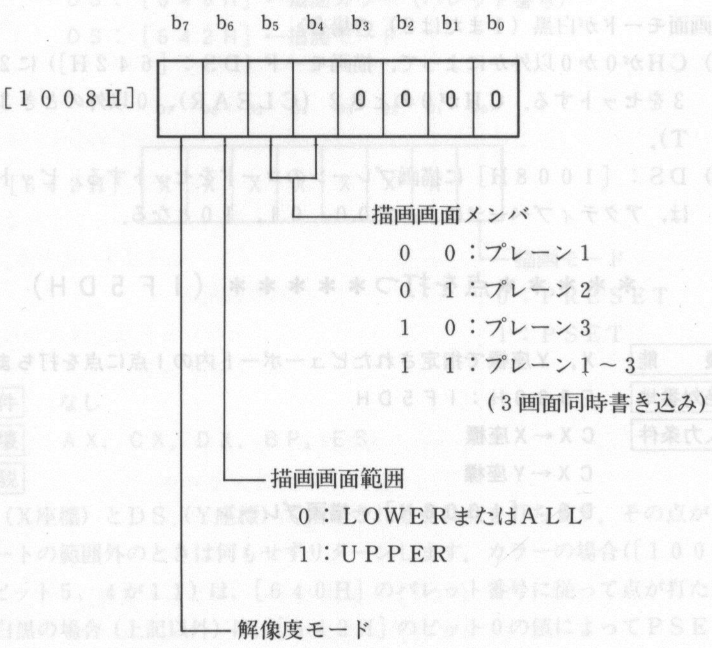
（画面モードがカラーのときのみ）

ＤＳ：［６４２Ｈ］←描画モード

２：ＣＬＥＡＲ

３：ＳＥＴ

ＤＳ：［１００８Ｈ］←描画プレーン



0：640×200

1：640×400

破壊　AX，CX

解説

CHに与えられた描画カラー，および，共通ワークエリアの画面モード，アクティブページの情報から描画モード，描画プレーンを求め，ワークエリアに格納します．画面モードがカラー（0または3）の場合は描画カラーも格納します．CHがFFH（省略）のときはフォアグラウンドカラーが使われます．

処理

〈画面モードがカラー（0または3）の場合〉

(1) DS：［640H］に描画カラー（CH）をセットする．CHがFFHの場合はフォアグラウンドカラー（DS：［623H］）を用いる．

(2) DS：［642H］に3（SET）をセットする．

(3) DS：［1008H］に描画プレーンのコードをセットする．ビット4，5は11（全プレーン）となる．

〈画面モードが白黒（1または2）の場合〉

(4) CHが0か0以外かによって，描画モード（DS：［642H］）に2または3をセットする．CHが0のとき2（CLEAR），0以外のとき3（SET）．

(5) DS：［1008H］に描画プレーンのコードをセットする．ビット4，5は，アクティブページによって00，01，10となる．

*****点を打つ*****（1F5DH）

機能　X，Y座標で指定されたビューポート内の1点に点を打ちます．

絶対番地　F990H：1F5DH

入力条件　CX←X座標

CX←Y座標

DS：［1008H］←描画プレーン

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
[１００８Ｈ]					X	X	X	X

b_5, b_4 ─ 描画画面ナンバ
- ０　０：プレーン１
- ０　１：プレーン２
- １　０：プレーン３
- １　１：プレーン１～３（カラーのとき）

b_6 ─ 描画画面範囲
- ０：ＬＯＷＥＲまたはＡＬＬ
- １：ＵＰＰＥＲ

ＤＳ：［６４０Ｈ］←描画カラー（パレット番号）
ＤＳ：［６４２Ｈ］←描画モード

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
[６４２Ｈ]	X	X	X	X	X	X	X	

b_0 ─ 描画モード
- ０：ＰＲＥＳＥＴ
- １：ＰＳＥＴ

出力条件　なし

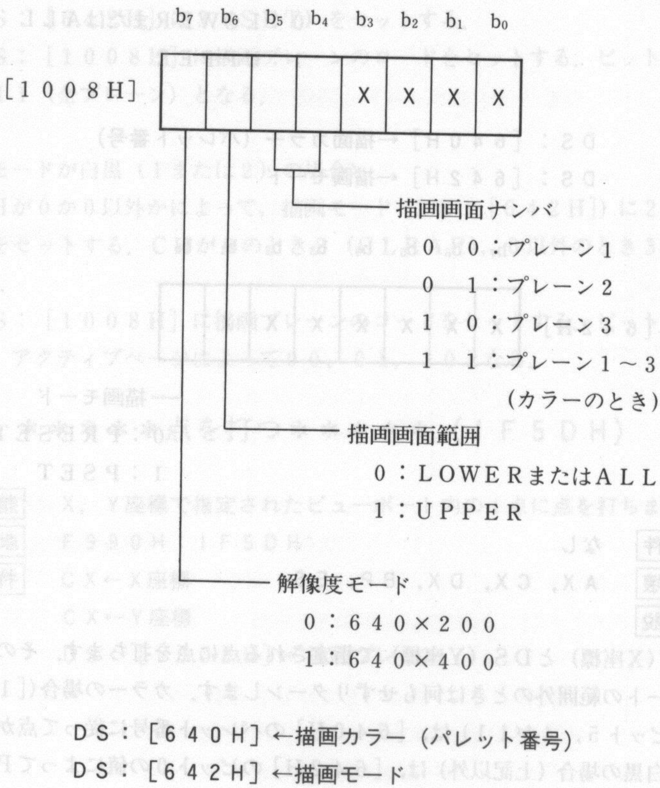
破　壊　ＡＸ，ＣＸ，ＤＸ，ＢＰ，ＥＳ

解　説

ＣＸ（X座標）とＤＳ（Y座標）で指定される点に点を打ちます．その点がビューポートの範囲外のときは何もせずリターンします．カラーの場合([１００８H］のビット５，４が１１）は，［６４０H］のパレット番号に従って点が打たれます．白黒の場合（上記以外）は，［６４２H］のビット０の値によってＰＳＥＴまたはＰＲＥＳＥＴとなります．

＊＊＊＊＊線を引く＊＊＊＊＊（2016H）

機　能	X，Y座標で指定された2点間に線を引きます．ただし，線の描画は，ビューポートの範囲外には行なわれません．
絶対番地	F990H：2016H
入力条件	DS：［10C0H～10C1H］←X座標1 DS：［10C2H～10C3H］←Y座標1 DS：［10C4H～10C5H］←X座標2 DS：［10C6H～10C7H］←Y座標2 DS：［10C8H～10C9H］←ラインスタイル DS：［1008H］←描画プレーン



DS：［640H］←描画カラー（パレット番号）

DS：［642H］←描画モード

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
［642H］	X	X	X	X	X	X	X	

b_0：描画モード
0：CLEAR（黒）
1：SET　　（白）

出力条件　なし

破　　壊　AX，CX，DX，BP，SI，DI，ES

解　　説

Ｘ１，Ｙ１からＸ２，Ｙ２へ線を引きます．Ｘ，Ｙは−３２７６８〜３２７６７の値をとることができますが，実際の描画はビューポート内のみに行なわれます．カラーの場合（［１００８Ｈ］のビット５，４が１１のとき）は，［６４０Ｈ］のパレット番号で線の色が決まります．白黒の場合（上記以外）は，［６４２Ｈ］のビット０の値によって黒または白の線となります．

処　　理

(1) 指定された線分とビューポートの枠との交点を求めることによって，実際の描画の座標を求める．
(2) ＢＩＯＳ直線の描画をコールする．（ＣＡＬＬ　２ＦＢ７Ｈ）

＊＊＊領域の塗りつぶし＊＊＊（２１Ｆ０Ｈ）

機　能　境界色で囲まれた領域を指定したタイルストリングで塗りつぶします．

絶対番地　Ｆ９９０Ｈ：２１Ｆ０Ｈ

入力条件　ＤＳ：［１０Ｃ０Ｈ～１０Ｃ１Ｈ］←塗りつぶし開始点のＸ座標
ＤＳ：［１０Ｃ２Ｈ～１０Ｃ３Ｈ］←塗りつぶし開始点のＹ座標
ＤＳ：［１０Ｃ４Ｈ］←１ラインあたりのタイルストリングのバイト数　白黒のとき　：１
カラーのとき：３
ＤＳ：［１０Ｃ５Ｈ］←境界色
カラーのとき　パレット番号
白黒のとき　　黒：０，白：１
ＤＳ：［１０Ｃ６Ｈ］←タイルストリング・ライン数
＝タイルストリング全体のバイト数を［１０Ｃ４Ｈ］の値で割った商
ＤＳ：［１０Ｃ８Ｈ～１０Ｃ９Ｈ］←タイルストリングの格納領域のセグメントアドレス
ＤＳ：［１０ＣＡＨ～１０ＣＢＨ］←タイルストリングの格納領域のオフセットアドレス
ＤＳ：［１０ＣＣＨ～１０ＣＤＨ］←作業領域最終オフセットアドレス
ＤＳ：［１０ＣＥＨ～１０ＣＦＨ］←作業領域先頭オフセットアドレス

出力条件　なし

破　壊　ＡＸ，ＢＸ，ＣＸ，ＤＸ，ＢＰ，ＤＩ，ＥＳ

解　説

ＬＩＯルーチン，ＧＰＡＩＮＴ１，ＧＰＡＩＮＴ２の下請け処理をします．描画開始点はビューポートの中にあることが必要です．作業領域はデータセグメント内にとります．１６バイト以上必要です．

親ルーチン

ＬＩＯ　ＧＰＡＩＮＴ１
ＬＩＯ　ＧＰＡＩＮＴ２

X軸に平行な線の描画（2609H）

機　能	X軸に平行な線上の2点間を指定したタイルストリングで結びます.
絶対番地	F990H：2609H
入力条件	DS：[10C4H]←1ラインあたりのタイルストリングのバイト数　白黒のとき：1　カラーのとき：3

DS：[10C6H]←タイルストリング・ライン数
＝タイルストリング全体のバイト数を[10C4H]の値で割った商

DS：[10C8H～10C9H]←タイルストリングの格納領域のセグメントアドレス

DS：[10CAH～10CBH]←タイルストリングの格納領域のオフセットアドレス

DS：[10D1H]←描画プレーン

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
[10D1H]	X				X	X	X	X

b_5 b_4：描画画面ナンバ
- 0 0：プレーン1またはカラーのとき
- 0 1：プレーン2
- 1 0：プレーン3
- 1 1：許されない組み合わせ

b_6：描画画面範囲
- 0：LOWERまたはALL
- 1：UPPER

DS：[10DCH～10DDH]←Y座標

DS：[10DEH～10DFH]←始点のX座標（X1）

DS：[10E0H～10E1H]←終点のX座標（X2）

入力条件　なし

破　　壊　ＡＸ，ＢＸ，ＣＸ，ＤＸ，ＢＰ，ＤＩ

解　　説

（Ｘ１，Ｙ）から，（Ｘ２，Ｙ）へタイルストリングを用いて線を引きます．［１０Ｃ４Ｈ］の値が１（白黒）のときは，［１０Ｄ１Ｈ］で指定された特定のプレーンに対してのみ描かれ，［１０Ｃ４Ｈ］の値が３（カラー）のときは，３つのプレーンに対して描かれます．カラーを指定したときは，［１０Ｄ１Ｈ］のビット５，４は００として下さい．

使われるタイルストリングは，タイルストリング・ライン数が１のときは，格納領域の先頭から，タイルストリング・ライン数が１より大きいときは，Ｙ座標の値によって異なります．ビューポートの上辺から順に繰り返し使った場合と同じになります．

Ｘ，Ｙはそれぞれ，次の関係を満たしていなくてはなりません．

０≦Ｘ≦Ｘ２≦６３９

０≦Ｙ≦１９９　（２００本モード）

or　３９９　（４００本モード）

本ルーチンは，ビューポートの影響を受けません．このルーチンは，塗りつぶしに使われます．

親ルーチン

ＬＩＯ　ＧＣＩＲＣＬＥ

＊＊＊画面にＰＵＴする＊＊＊（２６６９Ｈ）

機　能　格納領域にあるグラフィックパタンを画面に書き込みます．

絶対番地　Ｆ９９０Ｈ：２６６９Ｈ

入力条件　ＤＳ：［１００ＡＨ～１００ＢＨ］←書き込む領域の左上のＸ座標

ＤＳ：［１００ＣＨ～１００ＤＨ］←書き込む領域の左上のＹ座標

ＤＳ：［１００ＥＨ～１００ＦＨ］←Ｘ方向のドット数

ＤＳ：［１０１０Ｈ～１０１１Ｈ］←Ｙ方向のドット数

ＤＳ：［１０１２Ｈ］←格納データのカラー白黒フラグ

０：白黒

１：カラー

ＤＳ：［１０１３Ｈ］←描画モード

０：ＰＳＥＴ

１：ＰＲＥＳＥＴ

２：ＯＲ

３：ＡＮＤ

４：ＸＯＲ

ＤＳ：［１０１４Ｈ］←白の部分につける色（パレット番号）

ＤＳ：［１０１５Ｈ］←黒の部分につける色（パレット番号）

ＤＳ：［１０１６Ｈ～１０１７Ｈ］←格納領域セグメントベース

ＤＳ：［１０１８Ｈ～１０１９Ｈ］←格納領域オフセットアドレス

出力条件　なし

破　壊　ＡＸ，ＣＸ，ＤＸ，ＢＰ，ＳＩ，ＤＩ，ＥＳ

解　説

ＬＩＯルーチン，ＧＰＵＴ１，ＧＰＵＴ２の下請けを処理します．描画モードを０とすると，格納データがそのまま書き込まれ，１にすると反転して書き込まれます．２～４は画面上のデータと格納データとの演算結果が書き込まれます．

［１０１２Ｈ］のカラー白黒フラグは格納データがカラーの場合は［１０１４Ｈ］，［１０１５Ｈ］で白，黒の部分に対応する色を指定します．

親ルーチン

ＬＩＯ　ＧＰＵＴ１

ＬＩＯ　ＧＰＵＴ２

＊＊ＧＤＣの描画終了を待つ＊＊（３４８ＦＨ）

機　能　ＧＤＣのＦＩＦＯが空になり，かつ描画中でなくなるまで待ちます．

絶対番地　Ｆ９９０Ｈ：３４８ＦＨ

入力条件　なし

出力条件　なし

破　壊　ＡＸ

解　説

このルーチンはＣＰＵからＧＶＲＡＭに直接アクセスをするときに使われます．ＧＤＣとＣＰＵは同時に動作しますが，ＧＤＣが描画中にＣＰＵがＧＶＲＡＭにアクセスすると，ＧＤＣによる描画が正常に行なわれなくなるので，それに対処するため，ＧＤＣが描画中のときは，それが終了するまで待たなくてはいけません．このルーチンではＧＤＣのステータスを調べ，ＧＤＣのＦＩＦＯが空になり，かつ，描画中でなくなるまで待ってリターンします．

親ルーチン

ＬＩＯ　ＧＳＣＲＥＥＮ
ＬＩＯ　ＧＶＩＥＷ
ＬＩＯ　ＧＣＬＳ
ＬＩＯ　ＧＰＳＥＴ
ＬＩＯ　ＧＬＩＮＥ
ＬＩＯ　ＧＣＩＲＣＬＥ
ＬＩＯ　ＧＲＯＬＬ

＊＊矩形領域の塗りつぶし＊＊（３４９ＦＨ）

機　　能　Ｘ，Ｙ座標で指定された２点を対角線とする矩形領域を単一色で塗りつぶします．

絶対番地　Ｆ９９０Ｈ：３４９ＥＨ

入力条件　ＤＳ：［６４８Ｈ～６４９Ｈ］←矩形領域の左上のＸ座標（Ｘ１）

ＤＳ：［６４ＡＨ～６４ＢＨ］←矩形領域の左上のＹ座標（Ｙ１）

ＤＳ：［６５６Ｈ～６５７Ｈ］←矩形領域の右下のＸ座標（Ｘ２）

ＤＳ：［６５８Ｈ～６５９Ｈ］←矩形領域の右下のＹ座標（Ｙ２）

ＤＳ：［１００８Ｈ］←描画プレーン

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
［１００８Ｈ］	X				X	X	X	X

b_5–b_4：描画画面ナンバ

０　０：プレーン１

０　１：プレーン２

１　０：プレーン３

１　１：プレーン１～３（カラーのとき）

b_6：描画画面範囲

０：ＬＯＷＥＲまたはＡＬＬ

１：ＵＰＰＥＲ

ＤＳ：［６４０Ｈ］←描画カラー（パレット番号）

ＤＳ：［６４２Ｈ］←描画モード

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
［６４２Ｈ］	X	X	X	X	X	X	X	

b_0：描画モード

出力条件 なし

破　壊 AX，CX，DX，BP，DI

解　説

（X1，Y1）と（X2，Y2）を対角線とする矩形領域を単一色で塗りつぶします．X，Yはそれぞれ，次の関係を満たしていなくてはなりません．

0≦X1≦X2≦639

0≦Y1≦Y2≦199　（200本モード）

or399　（400本モード）

本ルーチンは，ビューポートの影響を受けません．カラーの場合（［1008H］のビット5，4が11のとき）は，［640H］のパレット番号で示される色で塗りつぶされます．白黒の場合（上記以外）は，［642H］のビット0の値によって黒または白で塗りつぶされます．

＊＊X軸に平行な線の描画（単一プレーン）＊＊

機　能 X軸に平行な線上の2点間を指定したラインスタイルで結びます．このルーチンによる描画は，指定した1つのプレーンに対してのみ行なわれます．

絶対番地 F990H：352FH

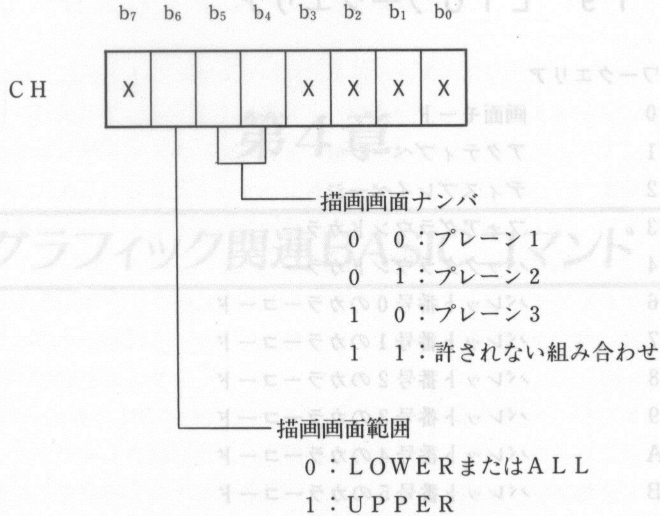
入力条件 DS：［648H～649H］←始点のX座標（X1）

DS：［64AH～64BH］←Y座標

DS：［656H～657H］←終点のX座標（X2）

DS：［660H～661H］←ラインスタイル

CH←描画プレーン



出力条件　なし

破　壊　AX，CX，DX，BP，DI

解　説

（X1，Y）から（X2，Y）へ線を引きます．このときのラインスタイルは，DS：［660H～661H］の内容が使われます．X，Yはそれぞれ，次の関係を満たしていなくてはなりません．

0≦X1≦X2≦639

0≦Y≦199　（200本モード）

or399　（400本モード）

本ルーチンは，ビューポートの影響を受けません．このルーチンは，塗りつぶしに使われます．

親ルーチン

＊矩形領域の塗りつぶし　（349EH）

＊X軸に平行な線の描画　（2609H）

３.１９　ＬＩＯワークエリア

共通ワークエリア

６２０	画面モード
６２１	アクティブページ
６２２	ディスプレイページ
６２３	フォアグラウンドカラー
６２４	バックグラウンドカラー
６２６	パレット番号０のカラーコード
６２７	パレット番号１のカラーコード
６２８	パレット番号２のカラーコード
６２９	パレット番号３のカラーコード
６２Ａ	パレット番号４のカラーコード
６２Ｂ	パレット番号５のカラーコード
６２Ｃ	パレット番号６のカラーコード
６２Ｄ	パレット番号７のカラーコード
６２Ｅ－６２Ｆ	ビューポートの左上のＸ座標
６３０－６３１	ビューポートの左上のＹ座標
６３２－６３３	ビューポートの右下のＸ座標
６３４－６３５	ビューポートの右下のＹ座標
６３６	表示バンク
６３７	描画バンク

ＢＩＯＳへの引数エリア

６４０	描画カラー
６４１	ボーダーカラー
６４２	描画モード
６４４－６４７	カラーパレット

第4章

グラフィック関連BASICコマンド

3．19　LIOワークエリア

共通ワークエリア

620	画面モード
621	アクティブページ
622	ディスプレイページ
623	フォアグラウンドカラー
624	バックグラウンドカラー
626	パレット番号0のカラーコード
627	パレット番号1のカラーコード
628	パレット番号2のカラーコード
629	パレット番号3のカラーコード
62A	パレット番号4のカラーコード
62B	パレット番号5のカラーコード
62C	パレット番号6のカラーコード
62D	パレット番号7のカラーコード
62E－62F	ビューポートの左上のX座標
630－631	ビューポートの左上のY座標
632－633	ビューポートの右下のX座標
634－635	ビューポートの右下のY座標
636	表示バンク
637	描画バンク

BIOSへの引数エリア

640	描画カラー
641	ボーダーカラー
642	描画モード
644－647	カラーパレット

４．０　ＢＡＳＩＣコマンドの概要

本章では，ＢＡＳＩＣコマンドの解析を扱います．ＰＣ－９８００シリーズには，N88－ＢＡＳＩＣが内蔵されており，高度なグラフィクスを実現しています．ＰＣ－９８００シリーズのソフトウェア構造の中では，ＢＡＳＩＣインタプリタはその頂点にあり，ＢＡＳＩＣからは直接ハードウェアは扱わず，実際の描画はＬＩＯに指示を出すことによって行なっています．

ＢＡＳＩＣには，ＬＩＯになかったものとして，ワールド座標，ＬＰ（Last referenced Point）などの概念が加わっており，ＢＡＳＩＣインタプリタの主な処理は，コマンドの解釈とワールド座標からスクリーン座標への変換，相対座標から実座標への変換などです．

本章では，ＢＡＳＩＣコマンドの働きをより深く知ること，インタプリタの処理を理解してコマンドの機能の変更，拡張，さらには，新しいＢＡＳＩＣコマンドを作れるような知識を持つことを目的とし，主なコマンドには詳細なフローチャートを付けてインタプリタの処理がわかるようにし，さらにそこで使われているほとんどの内部サブルーチンについて解説しました．これは，読者がインタプリタの解読を行なう際，または，新しいコマンドを作ろうとする際には，非常に役立つでしょう．

コマンド	座標系	対応するＬＩＯ	ＬＰ最終値
ＣＩＲＣＬＥ	ワールド	ＧＣＩＲＣＬＥ	円の中心
ＣＬＳ		ＧＣＬＳ	ビューポートの左上角
［1］ＣＯＬＯＲ		ＧＣＯＬＯＲ１	変化なし
［2］		ＧＣＯＬＯＲ２	変化なし
ＤＲＡＷ	ワールド		サブコマンド列による
ＧＥＴ	スクリーン	ＧＧＥＴ	領域の右下角
ＬＩＮＥ	ワールド	ＧＬＩＮＥ	直線の終点
ＭＡＰ	両方		変化なし
［1］ＰＡＩＮＴ	ワールド	ＧＰＡＩＮＴ１	変化なし
［2］ＰＡＩＮＴ	ワールド	ＧＰＡＩＮＴ２	変化なし
ＰＯＩＮＴ	ワールド		指定座標
［1］ＰＯＩＮＴ（関数）	両方		変化なし
［2］ＰＯＩＮＴ（関数）	スクリーン	ＧＰＯＩＮＴ	変化なし
ＰＲＥＳＥＴ	ワールド	ＧＰＳＥＴ	指定座標
ＰＳＥＴ	ワールド	ＧＰＳＥＴ	指定座標
ＰＵＴ	スクリーン	ＧＰＵＴ１，２	領域の左上角
ＲＯＬＬ	スクリーン	ＧＲＯＬＬ	変化なし
ＳＣＲＥＥＮ		ＧＳＣＲＥＥＮ	画面の左上角
ＶＩＥＷ	スクリーン	ＧＶＩＥＷ	ビューポートの左上角
ＶＩＥＷ（関数）	スクリーン		変化なし
ＷＩＮＤＯＷ	ワールド		ビューポートの左上角
ＷＩＮＤＯＷ（関数）	ワールド		変化なし

4．1　CIRCLE

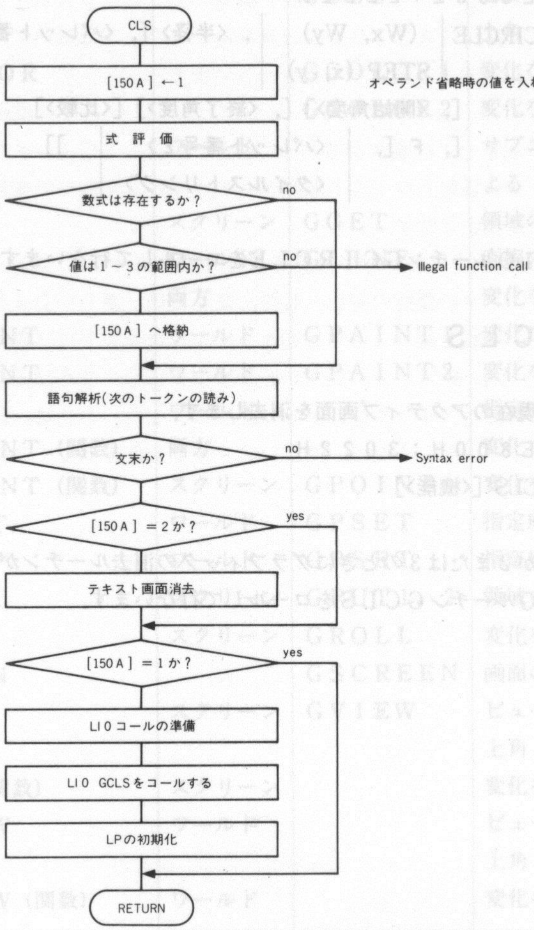
機　　能	円，楕円，円弧，扇形の描画，塗りつぶしを行なう（塗りつぶしは，ＰＣ－９８０１Ｅ／Ｆ／Ｍに限られる）．
絶対番地	Ｅ８００Ｅ：２Ｄ３２Ｈ
書　　式	CIRCLE { (Wx, Wy) / STEP (x, y) } ，〈半径〉［，〈パレット番号１〉］ ［，〈開始角度〉］［，〈終了角度〉］［〈比較〉］ ［，Ｆ［， { 〈パレット番号２〉 / 〈タイルストリング〉 } ］］
処　　理	

描画は，ＬＩＯルーチンＧＣＩＲＣＬＥをコールして行ないます．

4．2　CLS

機　　能	現在のアクティブ画面を消去します．
絶対番地	Ｅ８００Ｈ：３０２２Ｈ
書　　式	CLS［〈機能〉］
処　　理	

オペランドが2または3のときにグラフィックの消去ルーチンが使われます．消去は，ＬＩＯルーチンＧＣＬＳをコールして行ないます．


CLS
[150 A] ←1
オペランド省略時の値を入れる
式 評 価
数式は存在するか？
no
値は１～３の範囲内か？
no
Illegal function call
[150 A] へ格納
語句解析(次のトークンの読み)
文末か？
no
Syntax error
[150 A] ＝２か？
yes
テキスト画面消去
[150 A] ＝１か？
yes
LI０コールの準備
LI０ GCLSをコールする
LPの初期化
RETURN

4．3　COLOR

機　能　［1］画面の各部の色を指定します．

［2］カラーパレットを変更します．

［3］テキスト画面の文字に色をつけます．

絶対番地　E800H：3022H

書　式　［1］COLOR［〈ファンクションコード〉］［，〈バックグラウンドカラー〉］［，〈ボーダーカラー〉］［，〈フォアグラウンドカラー〉］

［2］COLOR=（〈パレット番号〉，〈カラーコード〉）

［3］COLOR@(X1,Y1)—(X2,Y2)［，〈ファンクションコード〉］

処　理

次に来るトークンによってコマンドが異なって解釈されます．“＝”の場合は［2］となり，“@”の場合は［3］となります．その他の場合が［1］で，はじめにここで処理ルーチンは分岐し，以下はそれぞれの処理ルーチンで行なわれます．

［1］の場合は，〈ファンクションコード〉を除く他のオペランドの処理は，LIOルーチンGCOLOR1をコールして行います．

［2］の場合は，LIOルーチンGCOLOR2をコールします．

4．4　DRAW

機　能　グラフィック描画サブコマンド列に従って，図形を描きます．このコマンドはPC－9801E／F／MのディスクBASICでのみ使用可能です．

絶対番地　E800H：3E5CH

書　式　DRAW〈文字列式〉

処　理

DIレジスタに33Hを代入して，システムコール（INT　C6H）でインタプリタRAMの起動を行ないます．

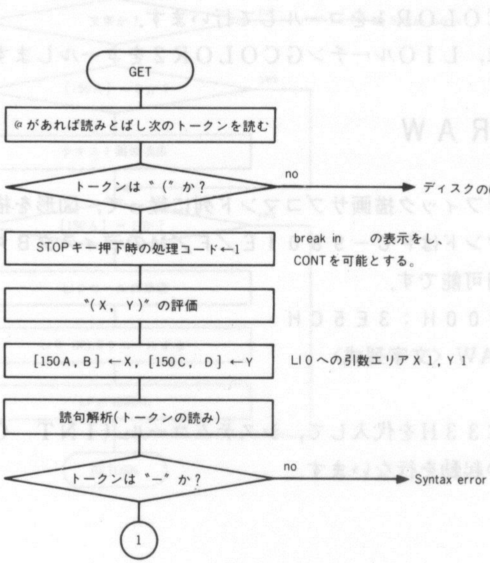
4．5　GET

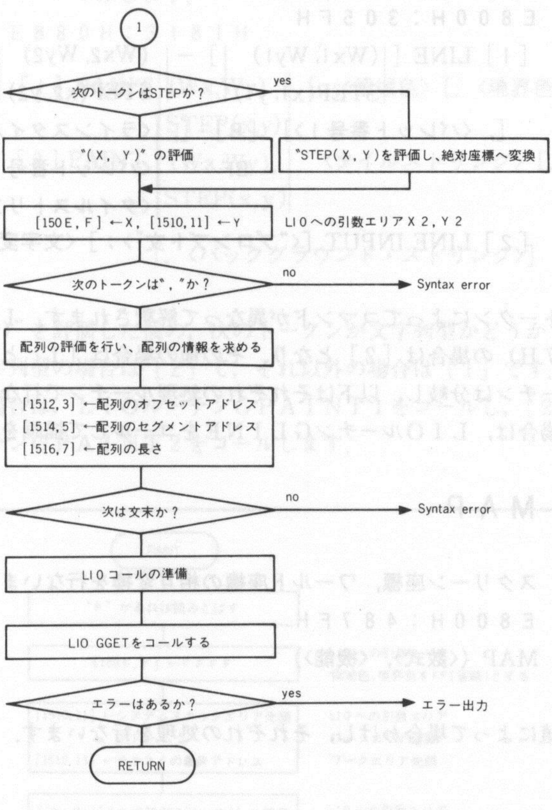
機　能	［1］ファイルの中のデータをバッファに読み込みます． ［2］画面上のグラフィックパタンを配列に読み込みます．
絶対番地	E800H：34A7H
書　式	［1］GET［#］〈ファイル番号〉［,〈数〉］ ［2］GET［@］(Sx1, Sy2)−|(Sx2, Sy2) / STEP(x, y)|,〈配列変数名〉［(〈要素〉)］

処　理

まず，先頭に“@”があったら読みとばします．次に来るトークンによってコマンドが異なって解釈されます．“(”の場合は［2］となり，その他の場合は［1］となります．ここで処理ルーチンは分岐し，以下はそれぞれの処理ルーチンで行なわれます．

［2］の場合は，LIOルーチンGGETをコールします．




1
次のトークンはSTEPか？
yes
"（X， Y）" の評価
"STEP（X，Y）を評価し、絶対座標へ変換
[150E，F] ←X，[1510，11] ←Y
LIOへの引数エリアX2，Y2
次のトークンは"，"か？
no
Syntax error
配列の評価を行い、配列の情報を求める
[1512，3] ←配列のオフセットアドレス
[1514，5] ←配列のセグメントアドレス
[1516，7] ←配列の長さ
次は文末か？
no
Syntax error
LIOコールの準備
LIO GGETをコールする
エラーはあるか？
yes
エラー出力
RETURN

４．６　ＬＩＮＥ

機　　能	［１］２点間に直線を引く，２点を対角線とする四辺形を描く，その内部を塗りつぶすなど．ただし，パレット番号２，タイルストリングによる塗りつぶしは，ＰＣ－９８０１Ｅ／Ｆ／Ｍに限られます． ［２］１行全体を入力します．
絶対番地	Ｅ８００Ｈ：３０５ＦＨ
書　　式	［１］LINE［｜(Wx1, Wy1) ／ STEP(x1, y1)｜］－｜(Wx2, Wy2) ／ STEP(x2, y2)｜ ［,〈パレット番号１〉］［,B ／ BF］［,｜〈ラインスタイル〉／〈パレット番号２〉／〈タイルストリング〉｜］ ［２］LINE INPUT［〈“プロンプト文”〉；］〈文字変数〉
処　　理	

次にくるトークンによってコマンドが異なって解釈されます．ＩＮＰＵＴ（内部コードＡ７H）の場合は［２］となり，その他の場合は［１］となります．ここで処理ルーチンは分岐し，以下はそれぞれの処理ルーチンで行なわれます．

［1］の場合は，ＬＩＯルーチンＧＬＩＮＥをコールして描画を行ないます．

４．７　ＭＡＰ

機　　能	スクリーン座標，ワールド座標の相互変換を行ないます．
絶対番地	Ｅ８００Ｈ：４８７ＦＨ
書　　式	MAP（〈数式〉，〈機能〉）
処　　理	

〈機能〉値によって場合わけし，それぞれの処理を行ないます．

4．8　PAINT

機　能　［1］指定された境界色で囲まれた領域を，指定された色で塗りつぶします．

［2］指定された境界色で囲まれた領域を，タイルパターンで塗りつぶします．

絶対番地　E880H：3181H

書　式

［1］PAINT　(Wx,Wy) ｜ STEP(x,y)　[,〈領域色〉[,〈境界色〉]]

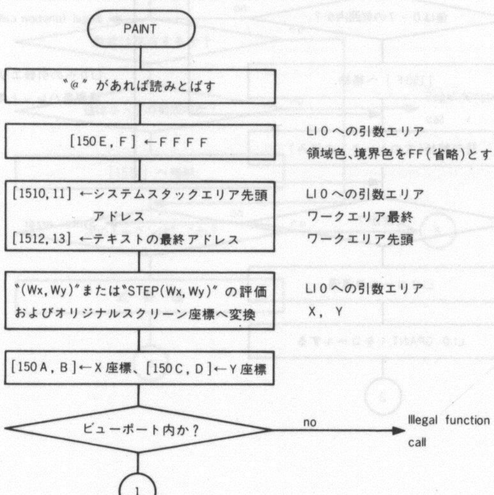
［2］PAINT　(Wx,Wy), ｜ STEP(x,y)　〈タイルストリング〉[,〈境界色〉]

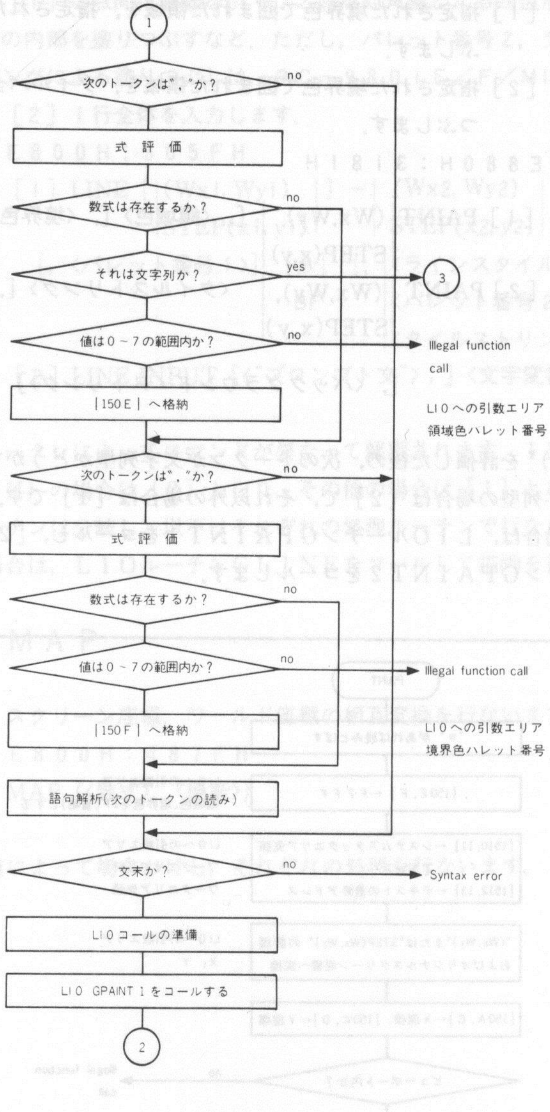
[,〈バックグラウンド・ストリング〉]

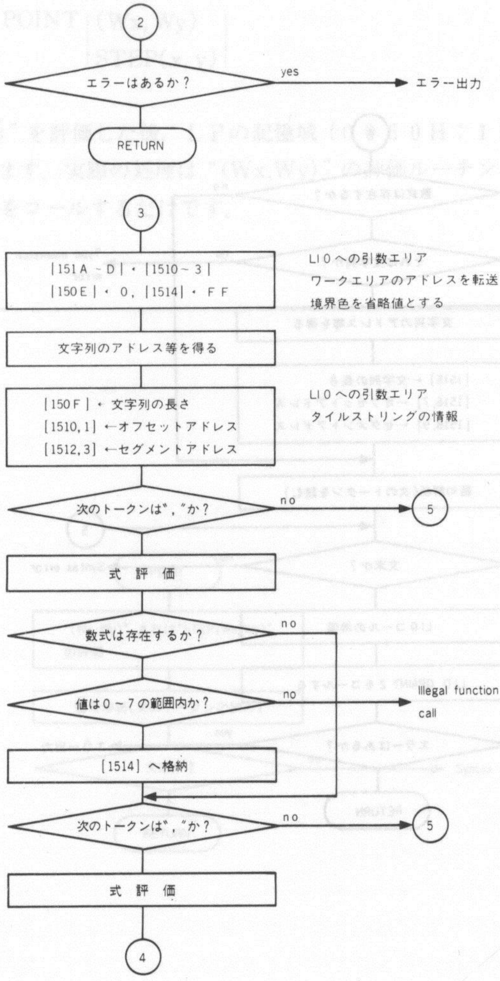
処　理

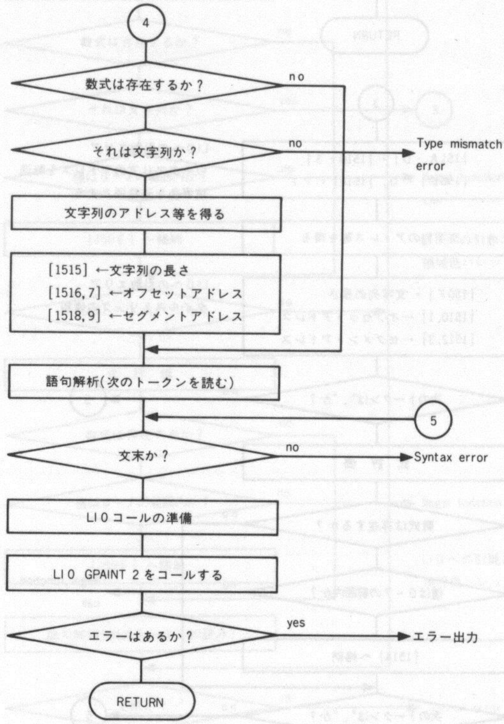
“(Wx,Wy)”を評価した後の，次のトークンが文字列型かどうかで処理が分岐します．文字列型の場合は［2］で，それ以外の場合は［1］です．

［1］の場合は，ＬＩＯルーチンＧＰＡＩＮＴ１をコールし，［2］の場合は，ＬＩＯルーチンＧＰＡＩＮＴ２をコールします．




1
次のトークンは","か?
no
式評価
数式は存在するか?
no
それは文字列か?
yes
3
値は0～7の範囲内か?
no
Illegal function call
[150E]へ格納
LIOへの引数エリア
領域色ハレット番号
次のトークンは","か?
no
式評価
数式は存在するか?
no
値は0～7の範囲内か?
no
Illegal function call
[150F]へ格納
LIOへの引数エリア
境界色ハレット番号
語句解析(次のトークンの読み)
文末か?
no
Syntax error
LIOコールの準備
LIO GPAINT1をコールする
2


2
エラーはあるか？
yes
エラー出力
RETURN
3
[151A～D]←[1510～3]
[150E]←0, [1514]←FF
LIOへの引数エリア
ワークエリアのアドレスを転送
境界色を省略値とする
文字列のアドレス等を得る
[150F]←文字列の長さ
[1510,1]←オフセットアドレス
[1512,3]←セグメントアドレス
LIOへの引数エリア
タイルストリングの情報
次のトークンは","か？
no
5
式評価
数式は存在するか？
no
値は0～7の範囲内か？
no
Illegal function call
[1514]へ格納
次のトークンは","か？
no
5
式評価
4

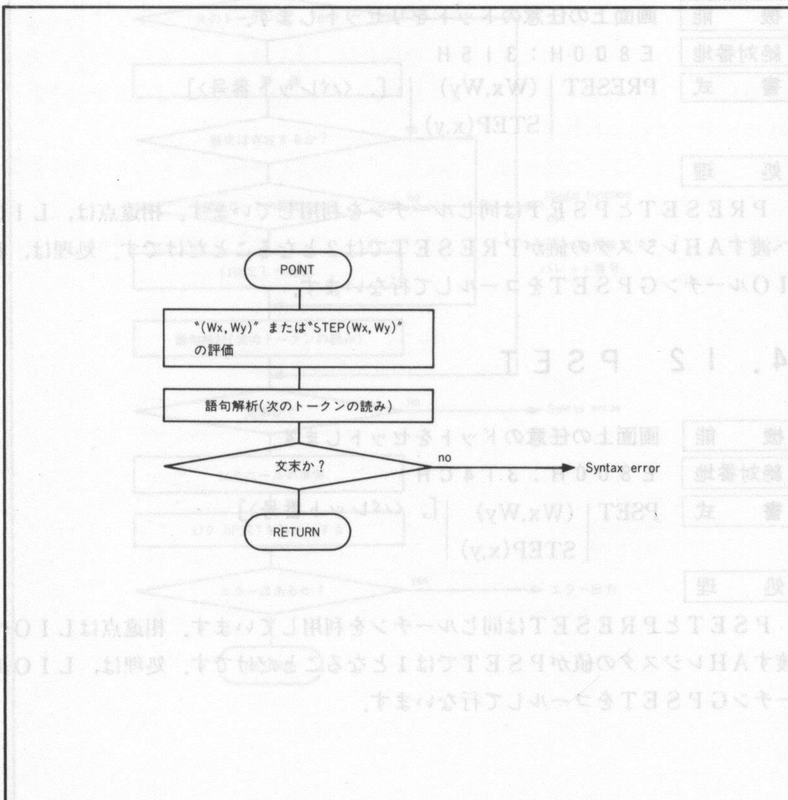

4
数式は存在するか？
no
それは文字列か？
no
Type mismatch error
文字列のアドレス等を得る
[1515] ←文字列の長さ
[1516,7] ←オフセットアドレス
[1518,9] ←セグメントアドレス
語句解析(次のトークンを読む)
5
文末か？
no
Syntax error
LIOコールの準備
LIO GPAINT 2 をコールする
エラーはあるか？
yes
エラー出力
RETURN

4．9　POINT

機　　能	Last Referenced Point　（ＬＰ）を変更します．
絶対番地	Ｅ８００Ｈ：３５０５Ｈ
書　　式	POINT (Wx, Wy) ／ STEP(x, y)

処　　理

“(Wx,Wy)”を評価した後，ＬＰの記憶域（００６０Ｈ：１５８８～１５８ＦＨ）に格納します．実際の処理は“(Wx,Wy)”の評価ルーチン（０Ｅ８００Ｈ：３６Ｃ５Ｈ）をコールするだけです．



4．10 POINT（関数）

機　能	［1］LPの値を与えます． ［2］指定された座標上のドットの色を与えます．
絶対番地	E800H：499CH
書　式	［1］POINT（<機能>） ［2］POINT（Sx,Sy）

処　理

最初の引数の後“，”があれば，［2］となり，それ以外は［1］となります．

4．11 PRESET

機　能	画面上の任意のドットをリセットします．
絶対番地	E800H：315H
書　式	PRESET ｜（Wx,Wy）/ STEP(x,y)｜［，<パレット番号>］

処　理

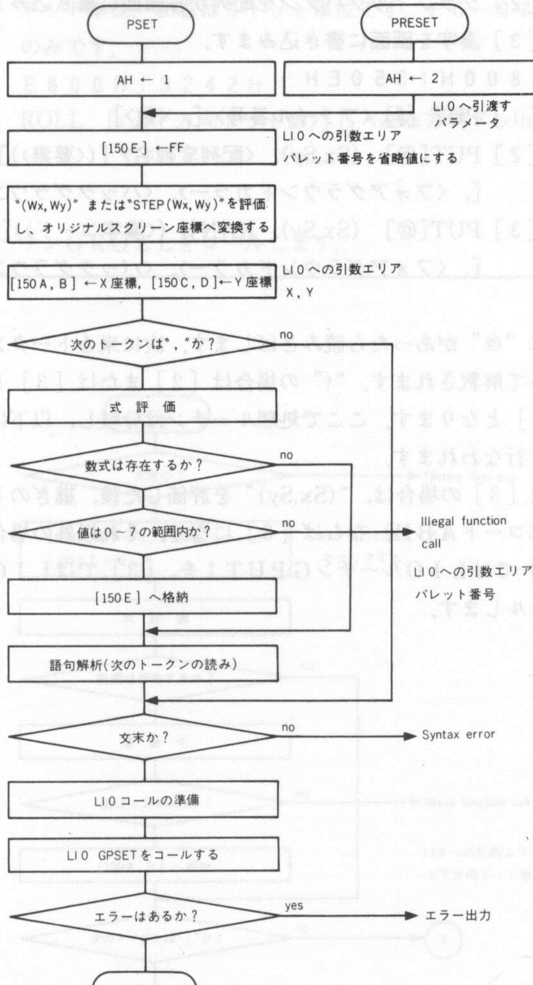
ＰＲＥＳＥＴとＰＳＥＴは同じルーチンを利用しています．相違点は，ＬＩＯへ渡すＡＨレジスタの値がＰＲＥＳＥＴでは2となることだけです．処理は，ＬＩＯルーチンＧＰＳＥＴをコールして行ないます．

4．12 PSET

機　能	画面上の任意のドットをセットします．
絶対番地	E800H：314CH
書　式	PSET ｜（Wx,Wy）/ STEP(x,y)｜［，<パレット番号>］

処　理

ＰＳＥＴとＰＲＥＳＥＴは同じルーチンを利用しています．相違点はＬＩＯへ渡すＡＨレジスタの値がＰＳＥＴでは1となることだけです．処理は，ＬＩＯルーチンＧＰＳＥＴをコールして行ないます．


PSET
PRESET
AH ← 1
AH ← 2
LIO へ引渡す
パラメータ
[150E] ←FF
LIO への引数エリア
パレット番号を省略値にする
"(Wx, Wy)" または"STEP(Wx, Wy)"を評価し、オリジナルスクリーン座標へ変換する
[150A, B] ←X座標, [150C, D]←Y座標
LIO への引数エリア
X, Y
次のトークンは","か？
no
式評価
数式は存在するか？
no
値は0～7の範囲内か？
no
Illegal function call
LIO への引数エリア
パレット番号
[150E] へ格納
語句解析(次のトークンの読み)
文末か？
no
Syntax error
LIO コールの準備
LIO GPSETをコールする
エラーはあるか？
yes
エラー出力
RETURN

4．13 PUT

機　能　［1］バッファの中のデータをファイルに書き出します．
［2］グラフィックパタンを配列から画面に書き込みます．
［3］漢字を画面に書き込みます．

絶対番地　E800H：350EH

書　式　［1］PUT［#］〈ファイル番号〉［，〈数〉］
［2］PUT［@］（Sx,Sy），〈配列変数名〉［(〈要素〉)］［，〈条件〉］
［，〈フォアグラウンドカラー〉，〈バックグラウンドカラー〉］
［3］PUT［@］（Sx,Sy），KANJI（〈漢字コード〉）［，〈条件〉］
［，〈フォアグラウンドカラー〉，〈バックグラウンドカラー〉］

処　理

まず，先頭に“@”があったら読みとばします．次に来るトークンによってコマンドが異なって解釈されます．“(”の場合は［2］または［3］となり，その他の場合は［1］となります．ここで処理ルーチンは分岐し，以下はそれぞれの処理ルーチンで行なわれます．

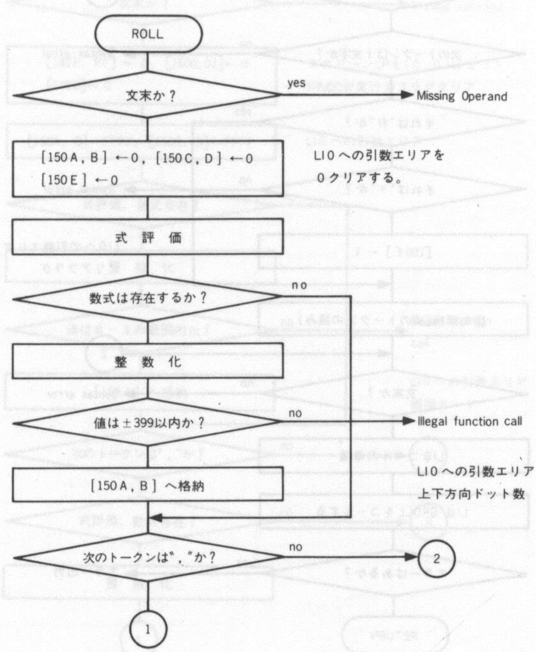
［2］または［3］の場合は，“(Sx,Sy)”を評価した後，継ぎのトークンがKANJI（内部コードABH）ならば［3］になり，それ以外の場合は［2］となります．［2］ではLIOルーチンGPUT1を，［3］ではLIOルーチンGPUT2をコールします．

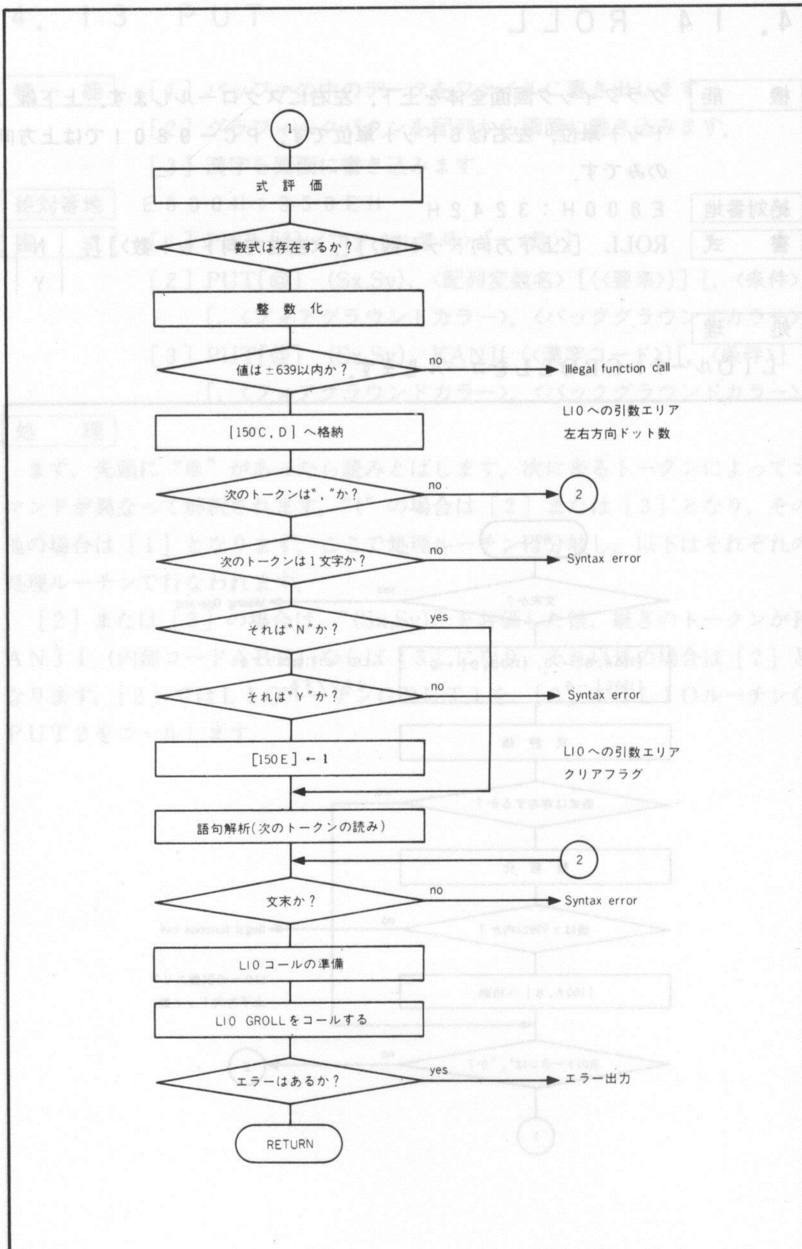
4．14　ROLL

機　能	グラフィック画面全体を上下，左右にスクロールします．上下は1ドット単位，左右は8ドット単位です．PC－9801では上方向のみです．
絶対番地	E800H：3242H
書　式	ROLL　[〈上下方向ドット数〉][,〈左右方向ドット数〉][, N / Y]

処　理

LIOルーチンGROLLをコールします．

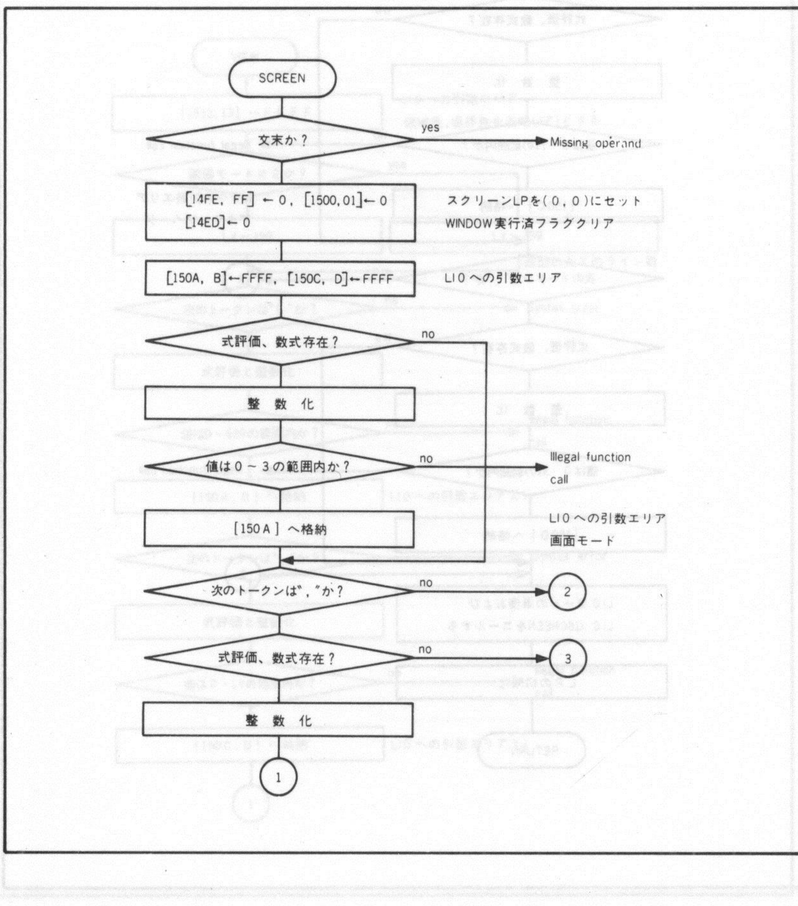


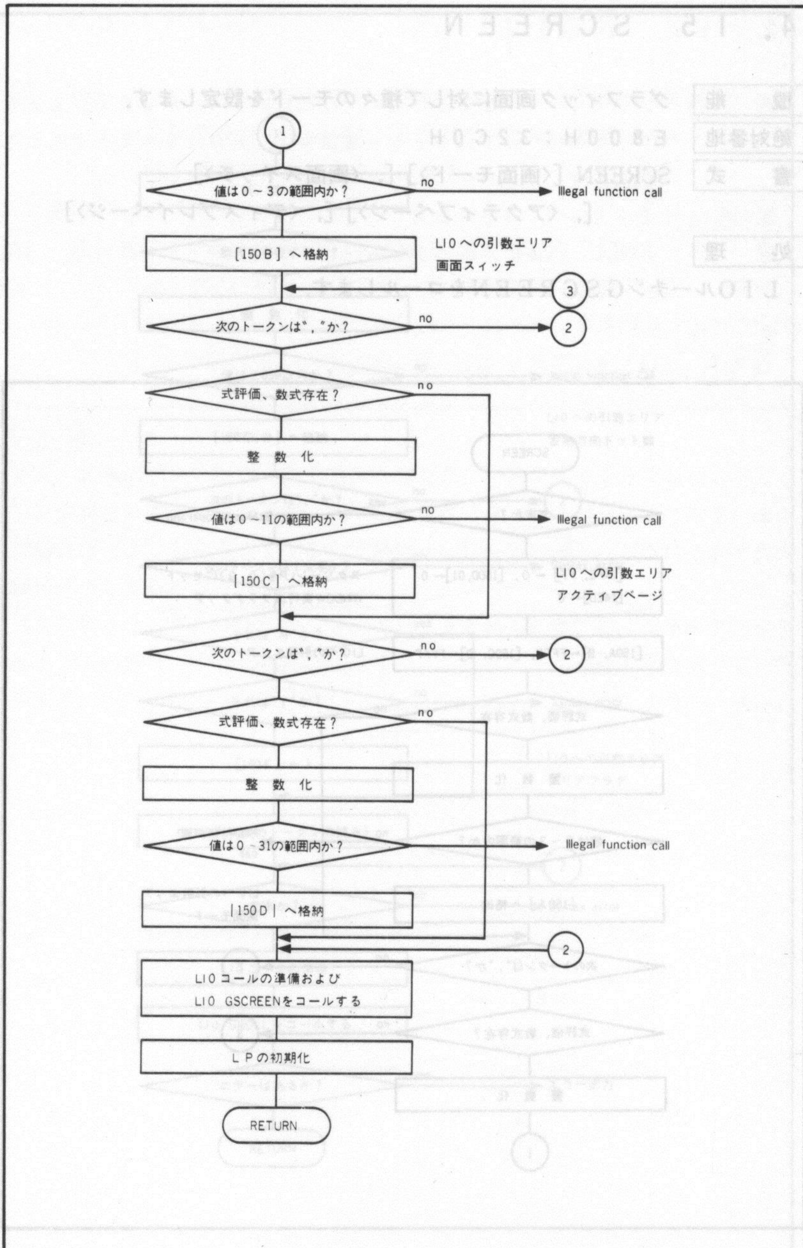

1
式評価
数式は存在するか？
整数化
値は±639以内か？
no
Illegal function call
[150C, D] へ格納
LIOへの引数エリア
左右方向ドット数
次のトークンは“,”か？
no
2
次のトークンは1文字か？
no
Syntax error
それは“N”か？
yes
それは“Y”か？
no
Syntax error
[150E] ← 1
LIOへの引数エリア
クリアフラグ
語句解析(次のトークンの読み)
2
文末か？
no
Syntax error
LIOコールの準備
LIO GROLLをコールする
エラーはあるか？
yes
エラー出力
RETURN

4．15 SCREEN

機　能	グラフィック画面に対して種々のモードを設定します．
絶対番地	E800H：32C0H
書　式	SCREEN［〈画面モード〉］［，〈画面スイッチ〉］［，〈アクティブページ〉］［，〈ディスプレイページ〉］
処　理	

LIOルーチンGSCREENをコールします．

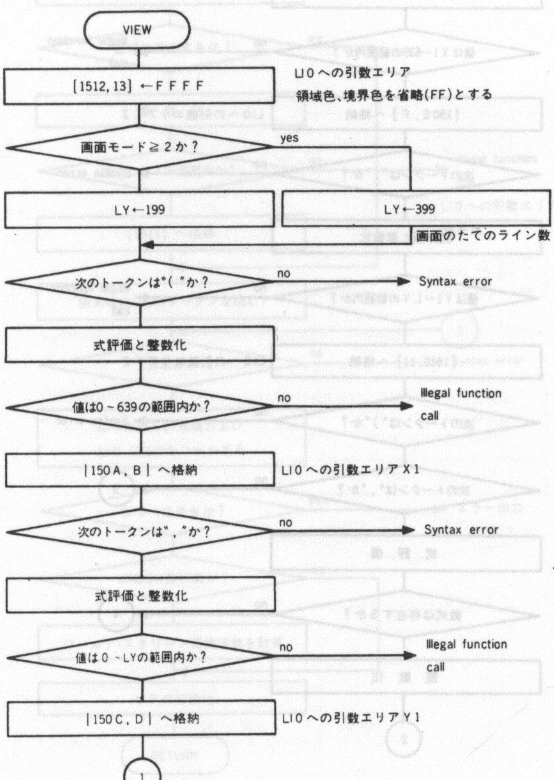


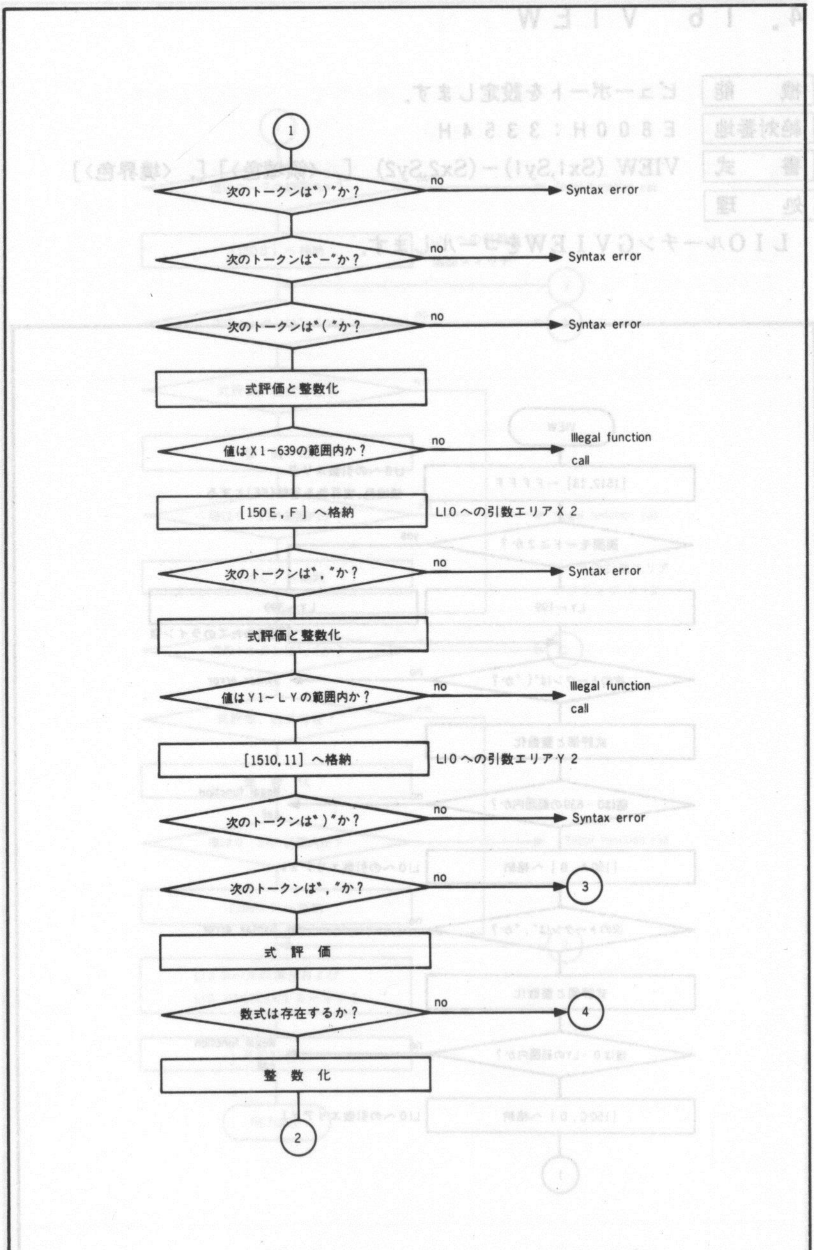

1
値は0～3の範囲内か？
no
Illegal function call
[150B] へ格納
LIOへの引数エリア
画面スイッチ
3
次のトークンは","か？
no
2
式評価、数式存在？
no
整 数 化
値は0～11の範囲内か？
no
Illegal function call
[150C] へ格納
LIOへの引数エリア
アクティブページ
次のトークンは","か？
no
2
式評価、数式存在？
no
整 数 化
値は0～31の範囲内か？
Illegal function call
[150D] へ格納
2
LIOコールの準備および
LIO GSCREENをコールする
LPの初期化
RETURN

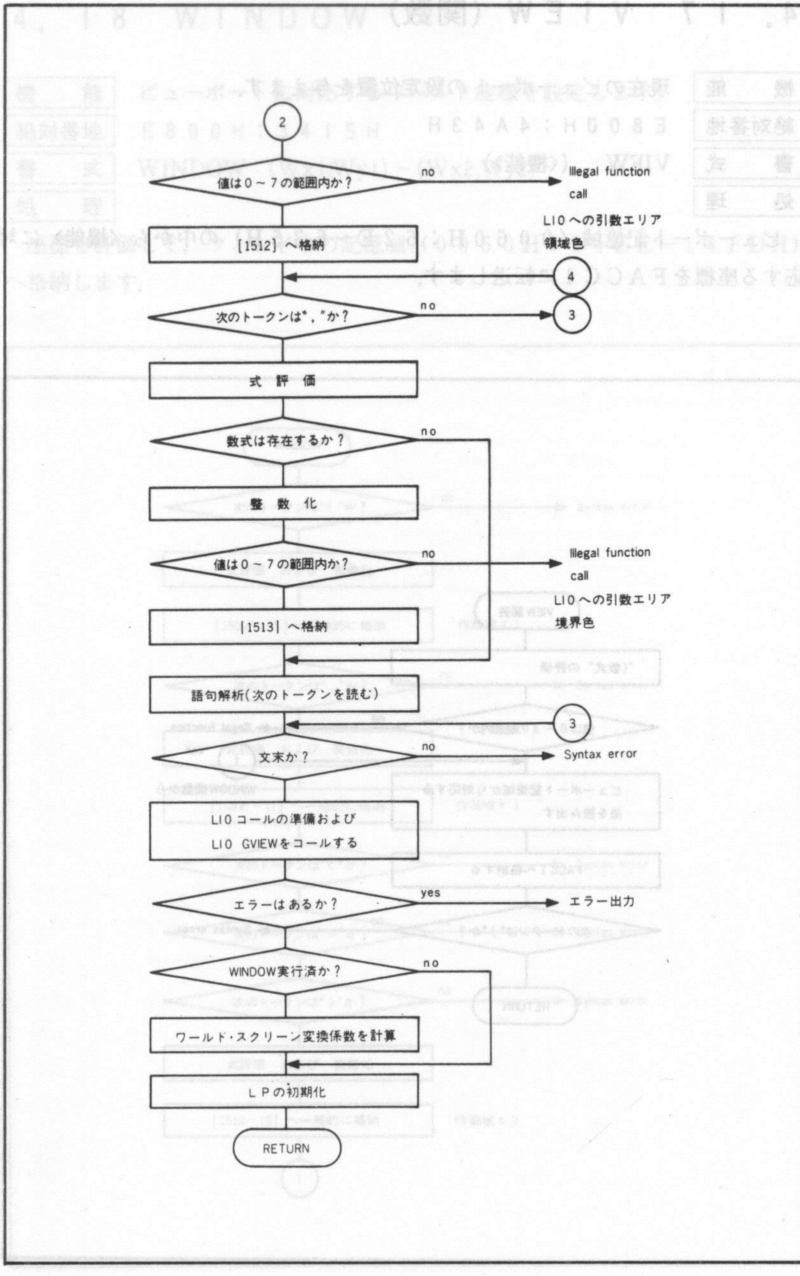
4．16 VIEW

機　能	ビューポートを設定します．
絶対番地	E800H：3354H
書　式	VIEW (Sx1,Sy1)－(Sx2,Sy2) [, <領域色>] [, <境界色>]
処　理	

ＬＩＯルーチンＧＶＩＥＷをコールします．




1
次のトークンは")"か?
no
Syntax error
次のトークンは"−"か?
no
Syntax error
次のトークンは"("か?
no
Syntax error
式評価と整数化
値はX1〜639の範囲内か?
no
Illegal function call
[150E, F] へ格納
LIOへの引数エリアX2
次のトークンは","か?
no
Syntax error
式評価と整数化
値はY1〜LYの範囲内か?
no
Illegal function call
[1510, 11] へ格納
LIOへの引数エリアY2
次のトークンは")"か?
no
Syntax error
次のトークンは","か?
no
3
式評価
数式は存在するか?
no
4
整数化
2


2
値は0～7の範囲内か？
no
Illegal function call
LIOへの引数エリア
領域色
[1512] へ格納
4
次のトークンは"，"か？
no
3
式評価
数式は存在するか？
no
整数化
値は0～7の範囲内か？
no
Illegal function call
LIOへの引数エリア
境界色
[1513] へ格納
語句解析（次のトークンを読む）
3
文末か？
no
Syntax error
LIOコールの準備および
LIO GVIEWをコールする
エラーはあるか？
yes
エラー出力
WINDOW実行済か？
no
ワールド・スクリーン変換係数を計算
LPの初期化
RETURN

4．17 VIEW（関数）

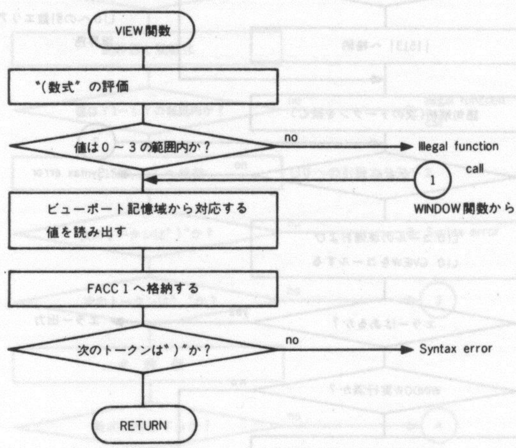
機　　能　現在のビューポートの設定位置を与えます．

絶対番地　E800H：4A43H

書　　式　VIEW　（<機能>）

処　　理

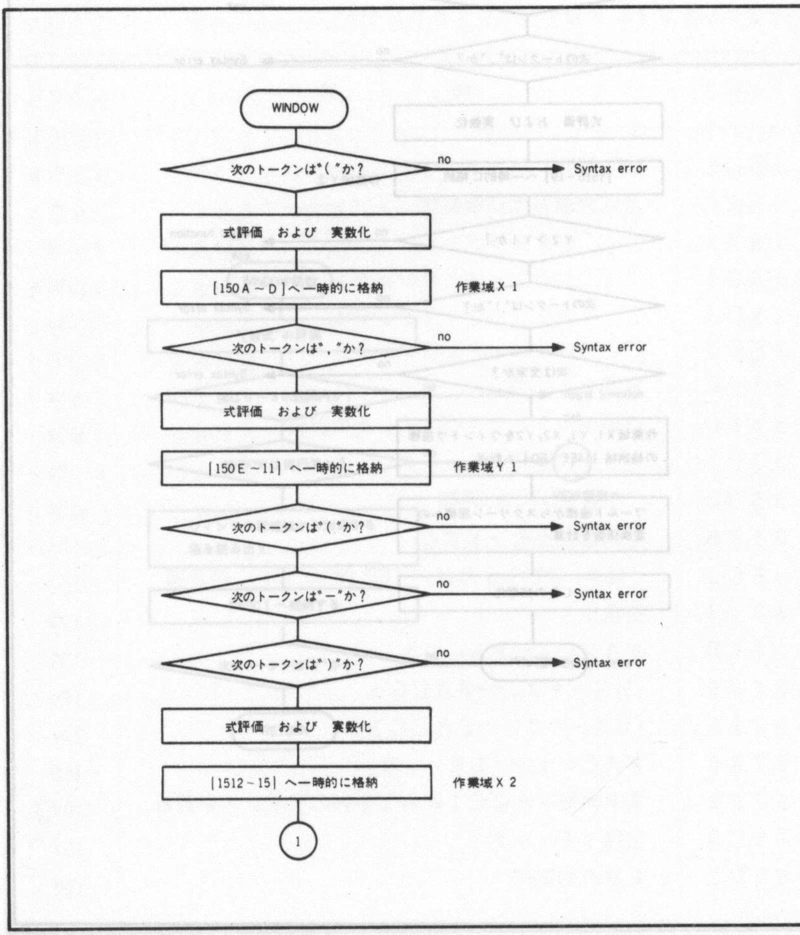
ビューポート記憶域（0060H：62E～635H）の中から〈機能〉に対応する座標をFACC1に転送します．

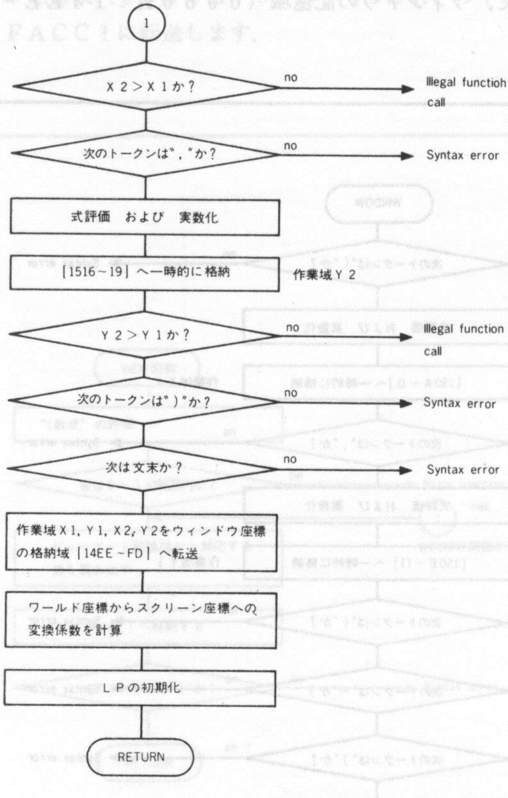


4．18 WINDOW

機　　能	ビューポートに対応するワールド座標を設定します．
絶対番地	E800H：3415H
書　　式	WINDOW （Wx1,Wy1）－（Wx2,Wy2）
処　　理	

座標を評価して，ウィンドウの記憶域（0060H：14EE～14FDH）へ格納します．

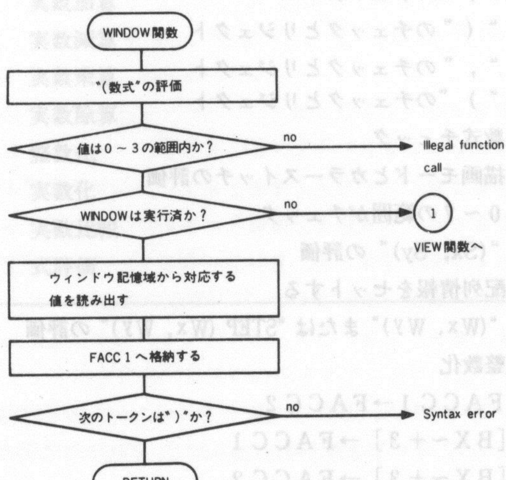



1
X 2 > X 1 か?
no
Illegal functioh call
次のトークンは", "か?
no
Syntax error
式評価　および　実数化
[1516~19] へ一時的に格納
作業域Y 2
Y 2 > Y 1 か?
no
Illegal function call
次のトークンは")"か?
no
Syntax error
次は文末か?
no
Syntax error
作業域X 1, Y 1, X 2, Y 2をウィンドウ座標の格納域 [14EE~FD] へ転送
ワールド座標からスクリーン座標への変換係数を計算
L P の初期化
RETURN

4．19 WINDOW（関数）

機　能	現在のウィンドウの設定位置を与えます。
絶対番地	E800H：4A68H
書　式	WINDOW　（〈機能〉）
処　理	

ウィンドウ記憶域（0060H：14EE～14FDH）の中に〈機能〉に対応する座標をFACC1に転送します．WINDOWが未実行の場合はVIEW関数のルーチンに移り，ビューポートの座標を得ることになります．



4．20　ＢＡＳＩＣ内部サブルーチン

ここでは，４．１から４．１９で解説したＢＡＳＩＣコマンドの処理で使われているほとんどのサブルーチンについて解説します．ここに含まれているサブルーチンはほとんどが文の解釈で使われます．読者が新しいＢＡＳＩＣコマンドを作るとき，既存のＢＡＳＩＣコマンドを改良しようとするときに利用すると便利です．次に全ルーチンの一覧表を掲げます．ページのところに“＊”をつけたルーチンは本シリーズ第１巻で解説されていますのでそちらを御参照下さい．

セグメントアドレスはすべてＥ８００Ｈです．

アドレス	機　　能	ページ
１ＢＣ２	語句解析	＊
１ＥＡＣ	数式評価および整数化	＊
１ＥＢ２	“，”のチェック	190
１ＥＤ１	“（”のチェックとリジェクト	190
１ＥＤＡ	“，”のチェックとリジェクト	191
１ＥＥ３	“）”のチェックとリジェクト	191
２１２Ｃ	数式チェック	＊
３５６Ｄ	描画モードとカラースイッチの評価	192
３５Ｄ１	０～７の範囲がチェック	193
３５ＤＤ	“(Sx, Sy)” の評価	193
３６７Ｂ	配列情報をセットする	194
３６Ｃ５	“(Wx, Wy)” または “STEP (Wx, Wy)” の評価	194
３７７４	整数化	195
３７７Ｂ	ＦＡＣＣ１→ＦＡＣＣ２	195
３７８８	［ＢＸ～＋３］→ＦＡＣＣ１	196
３７９５	［ＢＸ～＋３］→ＦＡＣＣ２	196
３７Ａ６	ＦＡＣＣ１→［ＢＸ～＋３］	196
３７Ｂ２	文字列型ＦＡＣＣ１から文字列のアドレスを得る	197
３７Ｃ９	文終了チェック	197
３７Ｄ２	ＬＰの初期化	198

3810	エラーチェックと出力	198
381A	ワールド・スクリーン変換比を求める	199
3907	@の読みとばしと“（ ”のチェック	200
3928	@の読みとばし	200
3935	“，”のチェック	200
393B	ＬＩＯコールの準備	201
3941	ビューポート内かどうかチェック	201
3D5D	“Syntax error”出力	*
3D62	“Illegal function call”出力	*
3D67	“Type mismactch”出力	*
4759	ＡＸ→ＦＡＣＣ１および“）”のチェック	202
491A	［ＢＸ～＋３］→ＦＡＣＣ１	196
493E	ＦＡＣＣ１→ＦＡＣＣ２	195
494E	ＦＡＣＣ１→［ＢＸ～＋３］	196
4B8C	“（数式”の評価	202
5627	実数加算	*
56BC	実数減算	*
56C7	実数乗算	*
5768	実数除算	*
5BEE	整数化	*
5C4D	実数化	*
5E14	実数比較	*
AEB2	式評価	*

＊＊＊区切り文字“,”のチェック＊＊＊

絶対番地	E800H：IEB2H
入力条件	DS←0060H
	AL←トークンの情報（中間コード等）
	BX←トークンのタイプ
出力条件	CF←0（トークンが“,”の場合）
	CF←I（それ以外の場合）
破　　壊	なし
解　　説	

このルーチンは，語句解析（1BC2）のコール後，“,”のチェックのためコールするものです．このルーチンを呼ぶ前にはAL，BXにトークンの情報等が格納されていなくてはなりません．ALにあるトークンが“,”であるなら，キャリーフラグをクリアし，それ以外の場合は，キャリーフラグをセットして，戻ります．

＊＊＊“(”のチェックとリジェクト＊＊＊

絶対番地	E800H：IEDIH
入力条件	DS←0060H
	0060H：[6EA，BH]←トークンの先頭アドレス
出力条件	0060H：[6E8，9H]←トークンの先頭アドレス
	0060H：[6EA，BH]←次のトークン先頭アドレス
エラー	Syntax error（トークンが“(”でない場合）
破　　壊	AX，BX，CX，DX，SI，DI，BP，ES
解　　説	

テキストポインタ（6EA，BH）の指すアドレスから語句解析を施し，トークンが“(”であるなら，そのまま呼び出し元へ戻り，それ以外の場合は，“Syntax error”を出力してインタプリタに制御を移します．

＊＊＊“,”のチェックとリジェクト＊＊＊

絶対番地	E800H：IEDAH
入力条件	DS←0060H
	0060H：［6EA，BH］←トークンの先頭アドレス
出力条件	0060H：［6E8，8H］←トークンの先頭アドレス
	0060H：［6EA，BH］←次のトークンの先頭アドレス
エラー	Syntax error （トークンが“,”でない場合）
破　壊	AX，BX，CX，DX，SI，DI，BP，ES
解　説	

テキストポイント（6EA，BH）の指すアドレスから語句解析を施し，トークンが“,”であるなら，そのまま呼び出し元へ戻り，それ以外の場合は，“Syntax error”を出力してインタプリタに制御を移します．

＊＊＊“)”のチェックとリジェクト＊＊＊

絶対番地	E800H：IEE3H
入力条件	DS←0060H
	0060H：［6EA，BH］←トークンの先頭アドレス
出力条件	0060H：［6E8，9H］←トークンの先頭アドレス
	0060H：［6EA，BH］←次のトークンの先頭アドレス
エラー	Syntax error （トークンが“)”でない場合）
破　壊	AX，BX，CX，DX，SI，DI，BP，ES
解　説	

テキストポインタ（6EA，BH）の指すアドレスから語句解析を施し，トークンが“(”であるなら，そのまま呼び出し元へ戻り，それ以外の場合は，“Syntax error”を出力してインタプリタに制御を移します．

＊＊＊条件（描画モード）とカラースイッチの評価＊＊＊

絶対番地	E800H：356DH
入力条件	DS←0060H 0060H：[6EA，BH]←トークンの先頭アドレス
出力条件	DS：[1514H]←描画モード値 DS：[1515H]←0（第2，3オペランドが省略された場合） 1（それ以外の場合） DS：[1516H]←第2オペランド値（フォアグラウンドカラー） DS：[1517H]←第3オペランド値（バックグラウンドカラー）
エラー	Syntax error（条件が定められたもの以外，第2，3オペランドが0～7の範囲外，第3オペランドで文が終了しない場合）
破　壊	AX，BX，CX，DX，SI，BP，ES
解　説	

テキストポインタ（6EA，BH）の指すアドレスから“[,〈条件〉][,〈フォアグラウンドカラー〉]”の形式のトークンの評価を行ない，DS：［1514H～1517H］に格納します．

条件と描画モードの値との対応は次のようになっています．

条　件	内部コード	描画モード値
PSET	C3H	0
PRESET	C4H	1
OR	F9H	2
AND	F8H	3
XOR	FAH	4

条件にこれ以外のトークンがあった場合，第2，第3のオペランドの評価値が0～7の範囲を超える場合，第3オペランドが文の終了でない場合には，“Syntax error”を出力してインタプリタに制御を移します．また，本ルーチンでは，LIOコールの準備（37C9H）も同時に行ないます．

＊＊＊整数化および0〜7の範囲チェック＊＊＊

絶対番地	E800H：35D1H
入力条件	DS←0060H
	FACC1←数値
	BCVAL←数値の型
出力条件	AX←整数化された数値
エラー	Illegal function call（数値が0〜7以外の場合）
破壊	AX, BX, DX, SI, DI
解説	

FACC1にある数値を整数化し，値が0〜7の範囲にあるかチェックする．範囲内にある場合は，AXレジスタにその値を入れて戻ります．範囲外の場合は，“Illigal function call”を出力してインタプリタに制御を移します．

＊＊＊“(Sx,Sy)”の評価＊＊＊

絶対番地	E800H：35DDH
入力条件	DS←0060H
	0060H：［6EA, BH］←トークンの先頭アドレス
出力条件	0060H：［6E8, 9H］←トークンの先頭アドレス
	0060H：［6EA, BH］←次のトークンの先頭アドレス
	AX←Sxの評価値（オリジナルスクリーン）
	BX←Syの評価値（オリジナルスクリーン）
	0060H：［14FE, FH］←Sxの評価値
	0060H：［1500, 1H］←Syの評価値
	0060H：［1588〜BH］←Sxのワールド座標への変換値
	0060H：［158C〜FH］←Syのワールド座標への変換値
エラー	Syntax error（トークンが“(Sx,Sy)”の形式になっていない場合）
	Over flow（Sx Syの評価値が2バイト整数の範囲を超えた場合）
	Illegal function call（Sx, Syがビューポート外の場合）
破壊	AX, BX, CX, DX, SI, BP, ES
解説	

テキストポインタ（6EA, BH）の指すアドレスから“(Sx,Sy)”の形式のトークンの評価を行い，Sxの値をAX，Syの値をBXレジスタにそれぞれ格納

して戻ります．またこの値でＬＰを更新します．スクリーンＬＰにはそのままで，ワールドＬＰには，ワールド座標に変換して格納されます．この形式になっていない場合は，“Syntax error”を出力してインタプリタに制御を移します．

＊＊＊配列名からアドレス等を得る＊＊＊

絶対番地	Ｅ８００Ｈ：３６７ＢＨ
入力条件	ＤＳ←００６０
	００６０Ｈ：［６ＥＡ，ＢＨ］←トークンの先頭アドレス
出力条件	００６０Ｈ：［６Ｅ８，９Ｈ］←トークンの先頭アドレス
	００６０Ｈ：［６ＥＡ，ＢＨ］←次のトークンの先頭アドレス
	ＡＸ←配列のオフセットアドレス
	ＢＸ←配列のセグメントアドレス
	ＣＸ←配列の長さ（バイト）
エラー	Syntax error（トークンが配列名でない場合）
破　　壊	ＡＸ，ＢＸ，ＣＸ，ＤＸ，ＳＩ，ＤＩ，ＢＰ，ＥＳ
解　　説	

テキストポインタ（６ＥＡ，ＢＨ）の指すアドレスにある配列名から，配列の先頭アドレスオフセット，セグメント，長さを求め，それぞれＡＸ，ＢＸ，ＣＸレジスタに格納して戻ります．トークンが配列名でない場合は，“Syntax error”を出力してインタプリタに制御を移します．

＊＊“(Wx,Wy)”または“STEP(Wx,Wy)”の評価＊＊

絶対番地	Ｅ８００Ｈ：３６Ｃ５Ｈ
入力条件	ＤＳ←００６０Ｈ
	００６０Ｈ：［６ＥＡ，ＢＨ］←トークンの先頭アドレス
出力条件	００６０Ｈ：［６Ｅ８，９Ｈ］←トークンの先頭アドレス
	００６０Ｈ：［６ＥＡ，ＢＨ］←次のトークンの先頭アドレス
	ＡＸ←Ｘ座標（オリジナルスクリーン座標へ変換後）
	ＢＹ←Ｙ座標（オリジナルスクリーン座標へ変換後）
	００６０Ｈ：［１４ＦＥ，ＦＨ］←Ｘ座標（同変換後）
	００６０Ｈ：［１５００，１Ｈ］←Ｙ座標（同変換後）

0060H：[1588～BH]←Wxの評価値

0060H：[158C～FH]←Wyの評価値

エラー　Syntax error（トークンが“(Wx,Wy)”または“STEP(Wx,Wy)”の形式になっていない場合）

Over flow（Wx,Wy）をオリジナルスクリーン座標へ変換後，値が2バイト整数の範囲を超えた場合）

破　壊　AX，BX，CX，DX，SI，DI，BP，ES

解　説

テキストポインタ（6EA，BH）の指すアドレスから“(Wx,Wy)”または“STEP(Wx,Wy)”の形式のトークンの評価を行ない，それぞれワールド座標からオリジナルスクリーン座標へ変換し，X座標をAX，Y座標をBXレジスタにそれぞれ格納して戻ります．またこの値でLPを更新します．スクリーンLPには変換後の値を，ワールドLPには，変換前の値が格納されます．この形式になっていない場合は，“Syntax error”を出力してインタプリタに制御を移します．

＊＊＊整数化＊＊＊

絶対番地　E800H：3774H

入力条件　DS←0060H

FACC1←数値

BCVAL←数値の型

出力条件　AX←整数化された数値

破　壊　AX，BX，DX，SI，DI

解　説

FACC1にある数値を整数化し，AXレジスタにその値を入れて戻ります．

＊＊＊FACC1→FACC2＊＊＊

絶対番地　E800H：377BH　および　E800H：493EH

入力条件　DS←0060H

FACC1←実数

出力条件　FACC2←同じ実数

破　壊　AX

解　説

FACC1（［141A～DH］）にある実数をFACC2（［1424～7］）へ転送します.

＊＊＊［BX～＋3］→FACC1＊＊＊

絶対番地	E800H：3788H および E800H：491AH
入力条件	DS←0060H BX←実数の先頭アドレス ［BX～＋3］←実数
出力条件	FACC1←同じ実数 BCVAL←4
破　壊	AX

解　説

［BX～＋3］にある実数をFACC1（［141A～DH］）へ転送します.

＊＊＊［BX～＋3］→FACC2＊＊＊

絶対番地	E800H：3795H
入力条件	DS←0060H BX←実数の先頭アドレス DS：［BX～＋3］←実数
出力条件	FACC2←同じ実数 BCVAL←4
破　壊	AX

解　説

［BX～＋3］にある実数をFACC2（［1424～7H］）へ転送します.

＊＊＊FACC1→［BX～＋3］＊＊＊

絶対番地	E800H：37A6H および E800H：494EH
入力条件	DS←0060H FACC1←実数 BX←転送先の先頭アドレス

出力条件　ＤＳ：［ＢＸ～＋３］←同じ実数

破　　壊　ＡＸ

解　　説

ＦＡＣＣ１（［１４１Ａ～ＤＨ］）にある実数を［ＢＸ～＋３］へ転送します．

＊＊文字列型ＦＡＣＣＩから文字列のアドレスを得る＊＊

絶対番地　Ｅ８００Ｈ：３７Ｂ２Ｈ

入力条件　ＤＳ←００６０Ｈ

　　　　　ＦＡＣＣＩ←文字列

出力条件　ＡＸ←文字列のオフセットアドレス

　　　　　ＢＸ←文字列のセグメントアドレス

　　　　　ＣＬ←文字列の長さ

破　　壊　ＡＸ，ＢＸ，ＣＸ，ＤＩ，ＥＳ

解　　説

ＦＡＣＣ１（［１４１Ａ～ＤＨ］）にある文字列（具体的には文字列のストリングディスクリプタのアドレス）から，文字列のオフセットアドレス，セグメントアドレス，長さを求め，それぞれ，ＡＸ，ＢＸ，ＣＬに格納してもどります．

＊＊＊文末チェック＊＊＊

絶対番地　Ｅ８００Ｈ：３７Ｃ９Ｈ

入力条件　ＤＳ←００６０Ｈ

　　　　　００６０Ｈ：［６ＥＡ，ＢＨ］←トークンの先頭アドレス

出力条件

エラー　Syntax error（文末でない場合）

破　　壊　ＡＸ，ＢＸ，ＣＸ，ＤＸ，ＳＩ，ＤＩ，ＢＰ，ＥＳ

解　　説

テキストポインタ（６ＥＡ，ＢＨ）の指すアドレスから語句解析を施し，そこが文末（具体的には０または“：”）なら，そのまま呼び出し元へ戻り，それ以外の場合は，“Syntax error”を出力してインタプリタに制御を移します．

＊＊＊ＬＰの初期化＊＊＊

絶対番地	Ｅ８００Ｈ：３７Ｄ２Ｈ
入力条件	ＤＳ←００６０Ｈ
出力条件	００６０Ｈ：［１４ＦＥ，ＦＨ］←スクリーンＬＰの初期値Ｘ
	００６０Ｈ：［１５００，１Ｈ］←スクリーンＬＰの初期値Ｙ
	００６０Ｈ：［１５８８～ＢＨ］←ワールドＬＰの初期値Ｘ
	００６０Ｈ：［１５８Ｃ～ＦＨ］←ワールドＬＰの初期値Ｙ
破　　壊	ＡＸ，ＢＸ
解　　説	

スクリーンおよびワールドＬＰの格納領域に初期値を代入します．スクリーンＬＰの初期値は，ビューポートの左上角のオリジナルスクリーン座標（ＶＩＥＷで設定した値）となります．ワールドＬＰの初期値は，ＷＩＮＤＯＷ文実行後か否かで変わり，ＷＩＮＤＯＷ文実行済フラグ（１４ＥＤＨ）を調べ，０（未実行）なら（０，０）が，０以外（実行済）なら．ビューポートの左上角のワールド座標（ＷＩＮＤＯＷで設定した値）となります．

＊＊＊エラーチェックとエラー出力＊＊＊

絶対番地	Ｅ８００Ｈ：３８１０Ｈ
入力条件	ＤＳ←００６０Ｈ
	ＡＨ←エラー番号
出力条件	なし
エ ラ ー	各エラー（ＡＨが０以外の場合）
破　　壊	なし
解　　説	

レジスタＡＨの内容が０以外の場合はその値に対応するエラーを出力してインタプリタに制御を移します．

＊＊＊ワールド・スクリーン変換比を求める＊＊＊

絶対番地	E800H：381AH
入力条件	DS←0060
	0060H：[62E, FH]←ビューポートの左上X座標 オリジナルスクリーン上（Sx1）
	0060H：[630, 1H]←ビューポートの左上Y座標 オリジナルスクリーン上（Sy1）
	0060H：[632, 3H]←ビューポートの右下X座標 オリジナルスクリーン上（Sx2）
	0060H：[634, 5H]←ビューポートの右下Y座標 オリジナルスクリーン上（Sy2）
	0060H：[14EE～F1H]←ビューポートの左上X座標 ワールド座標上（Wx1）
	0060H：[14F2～F5H]←ビューポートの左上Y座標 ワールド座標上（Wy1）
	0060H：[14F6～F9H]←ビューポートの右下X座標 ワールド座標上（Wx2）
	0060H：[14FA～FDH]←ビューポートの右下Y座標 ワールド座標上（Wy2）
出力条件	0060H：[1502～5H]←(Wx2−Wx1)／(Sx2−Sx1)
	0060H：[1506～9H]←(Wy2−Wy1)／(Sy2−Sy1)
破　壊	AX, BX, CX, DX, SI, DI, BP, ES
解　説	

ワールド座標上のビューポートの大きさとスクリーン座標上での大きさとの比を求めて格納領域に格納します．結果は実数型となります．

＊＊＊“@”の読みとばしと“(”のチェック＊＊＊

絶対番地	E800H：3907H
入力条件	DS←0060H
	0060H：[6EA，BH]←トークンの先頭アドレス
出力条件	0060H：[6E8，9H]←トークンの先頭アドレス
	0060H：[6EA，BH]←次のトークンの先頭アドレス
	CF←0（トークンが“(”の場合）
	CF←1（それ以外の場合）
破　壊	AX，BX，CX，DX，SI，BP，ES
解　説	

テキストポインタ（6EA，BH）の指すアドレスから語句解析を施し，トークンが“@”なら次のトークンに進み，それが“(”であるならキャリーフラグをクリアし，“(”でないならキャリーフラグをセットして，戻ります．

＊＊＊“@”の読みとばし＊＊＊

絶対番地	E800H：3928H
入力条件	DS←0060H
	0060H：[6EA，BH]←トークンの先頭アドレス
出力条件	0060H：[6E8，9H]←トークンの先頭アドレス
	0060H：[6EA，BH]←次のトークンの先頭アドレス
破　壊	AX，BX，CX，DX，SI，DI，BP，ES
解　説	

テキストポインタ（6EA，BH）の指すアドレスから語句解析を施し，トークンが“@”なら次のトークンに進みます．

＊＊＊“，”のチェック＊＊＊

絶対番地	E800H：3935H
入力条件	DS←0060H
	0060H：[6EA，BH]←トークンの先頭アドレス
出力条件	0060H：[6E8，9H]←トークンの先頭アドレス
	0060H：[6EA，BH]←次のトークンの先頭アドレス

ＣＦ←０（トークンが“，”の場合）

ＣＦ←１（それ以外の場合）

破　　壊　ＡＸ，ＢＸ，ＣＸ，ＤＸ，ＳＩ，ＤＩ，ＢＰ，ＥＳ

解　　説

テキストポインタ（６ＥＡ，ＢＨ）の指すアドレスから語句解析を施し，トークンが“，”であるなら，キャリーフラグをクリアし，それ以外の場合は，キャリーフラグをセットして，戻ります．

＊＊＊ＬＩＯコールの準備＊＊＊

絶対番地　Ｅ８００Ｈ：３９３ＢＨ

入力条件　ＳＳ←００６０Ｈ

出力条件　ＥＳ←００６０Ｈ

ＢＸ←１５０ＡＨ

破　　壊　なし

解　　説

ＬＩＯコールのため，ＥＳレジスタに６０Ｈを，ＢＸレジスタに１５０ＡＨ（ＬＩＯへの引数の先頭アドレス）を格納して戻ります．

＊＊＊ビューポート内かどうかのチェック＊＊＊

絶対番地　Ｅ８００Ｈ：３９４１Ｈ

入力条件　ＤＳ←００６０Ｈ

ＡＸ←Ｘ座標（オリジナルスクリーン）

ＢＸ←Ｙ座標（オリジナルスクリーン）

出力条件　なし

破　　壊　なし

エラー　Illegal function call（ｘ，ｙがビューポート外の場合）

解　　説

ＡＸ，ＢＸに与えられたＸ，Ｙ座標がビューポート内かどうかチェックし，範囲内なら，そのまま呼び出し元へ戻り，それ以外の場合は，“Illegal function call”を出力してインタプリタに制御を移します．

＊＊＊ＡＸ→ＦＡＣＣＩおよび“)”のチェック＊＊＊

絶対番地	Ｅ８００Ｈ：４７５９Ｈ
入力条件	ＤＳ←００６０Ｈ ００６０Ｈ：［６ＥＡ，ＢＨ］←トークンの先頭アドレス ＡＸ←整数
出力条件	００６０Ｈ：［６Ｅ８，９Ｈ］←トークンの先頭アドレス ００６０Ｈ：［６ＥＡ，ＢＨ］←次のトークンの先頭アドレス ＦＡＣＣＩ←同整数 ＢＣＶＡＬ←２
エラー	Syntax error（トークンが“(”でない場合）
破壊	ＡＸ，ＢＸ，ＣＸ，ＤＸ，ＳＩ，ＤＩ，ＢＰ，ＥＳ
解説	

ＡＸレジスタにある整数をＦＡＣＣ１へ格納した後，テキストポインタ（６ＥＡ，ＢＨ）の指すアドレスから語句解析を施し，トークンが“)”であるなら，そのまま呼び出し元へ戻り，それ以外の場合は，“Syntax error”を出力してインタプリタに制御を移します．

＊＊＊“(数式”の評価＊＊＊

絶対番地	Ｅ８００Ｈ：４Ｂ８ＣＨ
入力条件	ＤＳ←００６０Ｈ ００６０Ｈ：［６ＥＡ，ＢＨ］←トークンの先頭アドレス
出力条件	００６０Ｈ：［６Ｅ８，９Ｈ］←トークンの先頭アドレス ００６０Ｈ：［６ＥＡ，ＢＨ］←次のトークンの先頭アドレス ＦＡＣＣＩ←整数（数式の評価値）
エラー	Syntax error（先頭トークンが“(”でない場合）
破壊	ＡＸ，ＢＸ，ＣＸ，ＤＸ，ＳＩ，ＤＩ，ＢＰ，ＥＳ
解説	

テキストポインタ（６ＥＡ，ＢＨ）の指すアドレスから語句解析を施し，トークンが“(”でない場合は“Syntax error”を出力してインタプリタに制御を移します．“(”である場合は次の数式を評価し，整数化した後，値をＡＸレジスタに入れて戻ります．

４．２１　ＢＡＳＩＣワークエリア

ここではグラフィックに関連のあるワークエリアをまとめて掲げておきます．

（セグメント６０Ｈ）

アドレス	意　　味
１４ＥＤ	ＷＩＮＤＯＷ実行済フラグ（０：未実行，１：実行済）
１４ＥＥ－１４Ｆ１	ウィンドウの左上のＸ座標（実数）
１４Ｆ２－１４Ｆ５	ウィンドウの左上のＹ座標（実数）
１４Ｆ６－１４Ｆ９	ウィンドウの右下のＸ座標（実数）
１４ＦＡ－１４ＦＤ	ウィンドウの右下のＹ座標（実数）
１４ＦＥ－１４ＦＦ	スクリーン座標系のＬＰ　Ｘ座標（整数）
１５００－１５０１	スクリーン座標系のＬＰ　Ｙ座標（整数）
１５０２－１５０５	ワールド・スクリーン変換比　Ｘ方向（実数）
１５０６－１５０９	ワールド・スクリーン変換比　Ｙ方向（実数）
１５０Ａ－	ＬＩＯへの引数エリア
１５８８－１５８Ｂ	ワールド座標系のＬＰ　Ｘ座標（実数）
１５８Ｃ－１５８Ｆ	ワールド座標系のＬＰ　Ｙ座標（実数）

***AX→FACC[illegible]

絶対番地 [illegible]

入力条件 DS←0060H

0060H：[6EA，BH]←トークンの[illegible]

AX←整数

出力条件 0060H：[6E8，9H]←トークンの先頭アドレス

0060H：[illegible]

FACC[illegible]

BCVA[illegible]

エラー Syntax error [illegible]

破壊 AX，B[illegible]

解説

AXレジスタ[illegible]FACC[illegible]
A，BH)[illegible]
のまま呼び出[illegible] Syntax error [illegible]
プリタに制御を移します。

"(数式)"の評価

絶対番地 E800H：4B8CH

入力条件 DS←0060H

0060H：[6EA，BH]←トークンの先頭アドレス

出力条件 0060H：[6E8，9H]←トークンの先頭アドレス

0060H：[6EA，BH]←次のトークンの先頭アドレス

FACC1←整数（数式の評価値）

エラー Syntax error（先頭トークンが"("でない場合）

破壊 AX，BX，CX，DX，SI，DI，BP，ES

解説

テキストポインタ（6EA，BH）の指すアドレスから語句解析を施し，トークンが"("でない場合は"Syntax error"を出力してインタプリタに制御を移します。"("である場合は次の数式を評価し，整数化した後，値をAXレジスタに入れて戻ります。

第5章

応用プログラム集

5．0 N88－BASICグラフィック拡張コマンド

第１章で説明したように，ＰＣ９８００シリーズは，ＧＤＣというグラフィック描画機能をもったＬＳＩを搭載していますが，N88-ＢＡＳＩＣは，その機能を最大限に利用しているとは言えません．実際，N88-ＢＡＳＩＣが使用しているＧＤＣの描画機能は，ＬＩＮＥコマンドにおける直線の描画だけです．ＧＤＣの持つ円の描画，グラフィック文字の描画，四辺形の描画機能などがまったく使われていないのは残念です．そこで，ここではＧＤＣの機能を生かして，既存のＢＡＳＩＣコマンドを拡張した例を紹介します．

5．0．1 高速ＣＩＲＣＬＥ

○概要

ＧＤＣの円の描画機能を利用するものです．N88-ＢＡＳＩＣのＣＩＲＣＬＥコマンドは，プログラムでドットの位置を計算し，ＰＲＥＴで一つずつ点を打っています．円の対称性を利用して一度の計算で４点ずつ打っていますが，それでも，ＧＤＣの１ドット７００～８００ nsecのスピードに比べたら歯がたちません．しかしＧＤＣにも欠点はあります．例えば，

- 400本モードでの正円しか描けない．つまり惰円を描くことができない．また，200本モードでは正円も描くことができない．(正円とはＸ軸方向の半径とＹ軸方向の半径が等しい円のこと←惰円)
- 円弧を描く場合，ＧＤＣへ与えるパラメータの計算が面倒．また，ビューポートを設定して円の一部がビューポート外となる場合も同様に円弧となるので面倒．
- 塗りつぶしが同時にできない．後からＰＡＩＮＴで塗りつぶすとかえって時間がかかる．

などです．つまりN88-ＢＡＳＩＣのＣＩＲＣＬＥコマンドをＧＤＣだけでは実現できないこと．これがＧＤＣが使われなかった理由ではないかと思います．しかし，逆に言えば条件さえ満たせばＧＤＣを利用できるわけで，そのメリットは大きいといえます．

○仕様

本プログラムは，N88-ＢＡＳＩＣのＣＩＲＣＬＥコマンドと完全な互換性を実現

しました．以下の条件を満たしたときに，ＣＩＲＣＬＥコマンドは5～30倍速くなります．

満たさないときのスピードはほとんど変わりません．ＳＣＲＥＥＮコマンドで高速描画を指定すればさらに速くなります．

① ４００本モードであること．つまり，ＳＣＲＥＥＮ ２または３のとき．
② 開始角度，終了角度の指定がないこと．
③ 塗りつぶし (F) の指定がないこと．
④ 円はビューポートの範囲内におさまること．
⑤ 比率の指定のないこと．

○使用方法

本プログラムは機械語でできています．ダンプリストまたは，ソースリストをモニタで打ち込んで，ＢＳＡＶＥでフロッピディスクにセーブしておきます．本プログラムを使用するときは，ＣＬＥＡＲ文，ＤＥＦ ＳＥＧ文で機械語領域を定義してから，ＢＬＯＡＤ〝本プログラムの名前〞，Ｒとしてください．これ以後に現われるＢＡＳＩＣのＣＩＲＣＬＥコマンドは高速に実行されます．ＣＬＥＡＲ文，ＤＥＦ ＳＥＧ文，ＢＬＯＡＤ文は，プログラム中でも，ダイレクトモードでもどちらでもOKです．ＢＬＯＡＤ実行後は，ＤＥＦ ＳＥＧ文で定義を変更してもかまいません．

〈例〉
```
CLEAR,&H1F00
DEF  SEG=&H1F00
BLOAD "circle.bin",R
      ↑本プログラムにつけた名前
  .
  .
  .
CIRCLE (300,200),100
```

○プログラムの説明

本プログラムは，２つのプログラムからなっています．１つは高速にＣＩＲＣＬＥを実行する本体で，もう１つは，このプログラムをＢＡＳＩＣのＣＩＲＣＬＥルーチンと置き換えるプログラムです．ＢＬＯＡＤ〝ｃｉｒｃｌｅ．ｂｉｎ〞，Ｒを実行するとこの置き換えプログラムが走るのです．

〈置き換えのアルゴリズム〉

本プログラムのように，既存のコマンドと文法を全く変更せずに拡張する場合は，ＬＩＯを置き換えるという方法が使えます．今まで説明してきたように，ＢＡＳＩＣコマンドの多くは，ＬＩＯという下請けのルーチンをそれぞれ持っており，ＢＡＳＩＣコマンドの処理ルーチンでは文法の解析を行って，ＬＩＯへのパラメータを作成し，実際の描画はＬＩＯが受け持つというわけです．ＬＩＯは，ソフトウェア・インタラプト命令（ＩＮＴ）で呼び出していますから，インタラプトベクタテーブル（ＩＮＴ　ｎ命令の各々に対応するジャンプ先アドレスの書かれている表）を書きかえてやればいいのです．

本プログラムでは，ＬＩＯルーチンＧＣＩＲＣＬＥのＩＮＴ　Ａ８Ｈに対応するベクタテーブル（００００：２Ａ０～２Ａ３Ｈ）に高速ＣＩＲＣＬＥルーチンの先頭アドレスを書き込みます．

〈高速CIRCLEアルゴリズム〉

本プログラムはＬＩＯの置き換えですから，引数の受け渡しはＬＩＯに従います．実際の描画はＢＩＯＳの円弧描画ルーチンを呼び出して行いますが，そのまえに，先の条件を満たしているかのチェックを行い，満たしていない場合は，元のＬＩＯ　ＧＣＩＲＣＬＥに制御を移しています．

① 先に挙げたＧＤＣでできる条件のチェックをする．
② 条件を満たさない場合は，ＬＩＯ　ＧＣＩＲＣＬＥに制御を移す．
③ ＢＩＯＳの円弧描画をコールする．このルーチンは，一度には，８分の１の円弧しか描けないので，描画方向，描画開始点を変えて８回コールする．

○比較

どれだけスピードの向上が得られるかを計るため，次のようなＢＡＳＩＣプログラムをオリジナルの状態と本プログラムを動作させてからとを比較しました．このプログラムは，大中小の円をそれぞれ画面に５００個描き，それぞれの描画時間を計るものです．結果は右のように大円で４０倍，小円でも５倍速くなっています．円が大きいほど差が著しくなっていますが，これは小円では，円の描画にかかる時間が他の処理の時間に比べて少ないため全体の時間にすると差が小さく見えるためでしょう．純粋に円の描画時間だけを計れば５０倍以上と思われます．中円のときの画面の例を次に示します．

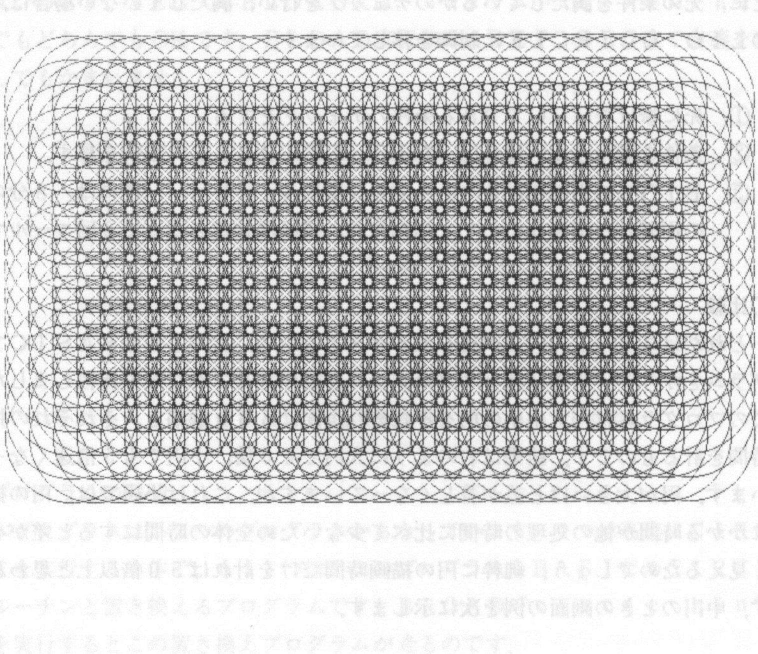
○スピード測定用サンプル

```
100 'CIRCLE DEMONSTRATION
110 SCREEN 3,1
120 CLS 2
130 'SMALL CIRCLE
140 TIME$="00:00:00"
150 FOR J=30 TO 370 STEP 25
160     FOR I=30 TO 610 STEP 25
170         CIRCLE (I,J),25
180     NEXT I
190 NEXT J
200 PRINT TIME$
210 CLS 2
220 'MEDIUM CIRCLE
230 TIME$="00:00:00"
240 FOR J=60 TO 340 STEP 20
250     FOR I=60 TO 580 STEP 20
260         CIRCLE (I,J),55
270     NEXT I
280 NEXT J
290 PRINT TIME$
300 CLS 2
310 'LARGE CIRCLE
320 TIME$="00:00:00"
330 FOR J=160 TO 240 STEP 8
340     FOR I=160 TO 480 STEP 8
350         CIRCLE (I,J),155
360     NEXT I
370 NEXT J
380 PRINT TIME$
```

○結果

```
run
00:00:12
00:00:29
00:01:28
Ok
clear ,&H1f00
Ok
def seg=&H1f00
Ok
run
00:00:02
00:00:03
00:00:04
Ok
```

○画面の例



○ソースリスト

```
                      ;
                      ;          HIGH SPEED GCIRCLE SUBROUTINE
                      ;          USING GDC
 02A0            INTA8    EQU 2A0H          ;VECTOR TABLE ADRS A8H
 0100            INT40    EQU 100H          ;VECTOR TABLE ADRS 40H

 062E            VIEWX1   EQU 62EH          ;VIEWPORT X1
 0630            VIEWY1   EQU 630H          ;VIEWPORT Y1
 0632            VIEWX2   EQU 632H          ;VIEWPORT X2
 0634            VIEWY2   EQU 634H          ;VIEWPORT Y2
 0620            SCRMOD   EQU 620H          ;SCREEN MODE
 0623            FORCLR   EQU 623H          ;FORE-GROUND COLOR
                      ;
                      ;       SET INTERUPT VECTOR TABLE
                      ;
                      ;       TRANSFER CONTENTS OF INTA8 TO INT40
                      ;       SET ADDRESS OF 'START' INTO INTA8
                      ;
0000 33C0        INIT:    XOR AX,AX
0002 8ED8                 MOV DS,AX
0004 A1A002               MOV AX,.INTA8
0007 A30001               MOV .INT40,AX
000A A1A202               MOV AX,.INTA8+2
000D A30201               MOV .INT40+2,AX
0010 C706A0021F00         MOV WORD PTR .INTA8,OFFSET START
0016 0E                   PUSH CS
0017 58                   POP AX
0018 A3A202               MOV .INTA8+2,AX
001B CF                   IRET
                      ;
                      ;       IN CASE THIS PROGRAM IS NOT SUITABLE,
                      ;       CALL ORIGINAL GCIRCLE ROUTINE
                      ;
001C CD40        TOROM:   INT 40H
001E CF                   IRET
                      ;
                      ;       HIGH SPEED GCIRCLE MAIN
                      ;
                      ;       CHECK RESTRICTIONS
                      ;
001F 807F0900    START:   CMP BYTE PTR 9[BX],0    ;ONLY CIRCLE?
0023 75F7         001C    JNE TOROM               ;NO
0025 8B5704               MOV DX,4[BX]            ;RX=RY?
0028 395706               CMP 6[BX],DX
002B 75EF         001C    JNE TOROM               ;NO
002D BF4006               MOV DI,640H
0030 8B07                 MOV AX,[BX]             ;GET X
0032 2BC2                 SUB AX,DX               ;X-R
0034 894508               MOV 8[DI],AX            ;STORE X-R
0037 3B062E06             CMP AX,.VIEWX1          ;X-R>=VIEWX1?
003B 7CDF         001C    JL TOROM                ;NO
003D 03C2                 ADD AX,DX
003F 03C2                 ADD AX,DX
0041 3B063206             CMP AX,.VIEWX2          ;X+R<=VIEWX2?
0045 7FD5         001C    JG TOROM                ;NO
0047 8B4702               MOV AX,2[BX]            ;GET Y
004A 89450A               MOV 0AH[DI],AX          ;STORE Y
004D 2BC2                 SUB AX,DX
004F 3B063006             CMP AX,.VIEWY1          ;Y-X>=VIEWY1?
0053 7CC7         001C    JL TOROM                ;NO
0055 03C2                 ADD AX,DX
0057 03C2                 ADD AX,DX
0059 3B063406             CMP AX,.VIEWY2          ;Y+X<=VIEWY2?
005D 7FBD         001C    JG TOROM                ;NO
005F A12006               MOV AX,.SCRMOD
0062 3C02                 CMP AL,2
0064 B403                 MOV AH,3
0066 7CB4         001C    JL TOROM
                      ;
                      ;       PREPARE PARAMETERS FOR BIOS CALL
                      ;
0068 B104                 MOV CL,4
006A D2E4                 SHL AH,CL
006C 8AEC                 MOV CH,AH
006E C7451A0000           MOV WORD PTR 1AH[DI],0  ;MASKING DOTS
0073 89551C               MOV 1CH[DI],DX          ;STORE RADIUS
```

```
0076 8A4708                 MOV AL,8[BX]            ;GET COLOR
0079 3CFF                   CMP AL,0FFH
007B 7503         0080      JNE NOTFF
007D A02306                 MOV AL,.FORCLR
0080 8805          NOTFF:   MOV [DI],AL             ;STORE COLOR
0082 C6450303               MOV BYTE PTR 3[DI],3    ;STORE DIRECTION
0086 C74520FFFF             MOV WORD PTR 20H[DI],0FFFFH   ;LINE STYLE
008B C6452804               MOV BYTE PTR 28H[DI],4  ;CIRCLE ID
008F 52                     PUSH DX
0090 B805B5                 MOV AX,0B505H           ;=1/SQR(2)*2^16
0093 F7E2                   MUL DX                  ;MEANS R/SQR(2)
0095 89550C                 MOV 0CH[DI],DX          ;STORE TOTAL DOTS
0098 5A                     POP DX
0099 8BDF                   MOV BX,DI
009B B448          LOOP:    MOV AH,48H              ;BIOS CALL
009D CD18                   INT 18H                 ;CIRCLE
009F 80470305               ADD BYTE PTR 3[BX],5    ;ROTATE DIRECTION
00A3 80670307               AND BYTE PTR 3[BX],7
00A7 B448                   MOV AH,48H              ;BIOS CALL
00A9 CD18                   INT 18H                 ;CIRCLE
00AB 80470305               ADD BYTE PTR 3[BX],5    ;ROTATE DIRECTION
00AF 80670307               AND BYTE PTR 3[BX],7
00B3 015708                 ADD 8[BX],DX            ;ADJUST DRAWING POINT
00B6 80F903                 CMP CL,3
00B9 7502         00BD      JNE NOT3
00BB F7DA                   NEG DX
00BD 01570A        NOT3:    ADD 0AH[BX],DX          ;ADJUST DRAWING POINT
00C0 FEC9                   DEC CL
00C2 75D7         009B      JNE LOOP
00C4 32E4                   XOR AH,AH
00C6 CF                     IRET
                   ;
                            END
```

○ダンプリスト

```
0000 33 C0 8E D8 A1 A0 02 A3 00 01 A1 A2 02 A3 02 01   3ﾀ■ﾘ。  ｣  .｢ ｣
0010 C7 06 A0 02 1F 00 0E 58 A3 A2 02 CF CD 40 CF 80   ﾇ      X｣｢ ﾏﾍ@ﾏ_
0020 7F 09 00 75 F7 8B 57 04 39 57 06 75 EF BF 40 06     u ■ W 9W u\ｿ@
0030 8B 07 2B C2 89 45 08 3B 06 2E 06 7C DF 03 C2 03   ■ +ﾂ E ; . |ﾟ ﾂ
0040 C2 3B 06 32 06 7F D5 8B 47 02 89 45 0A 2B C2 3B   ﾂ; 2  ﾕ■G E +ﾂ;
0050 06 30 06 7C C7 03 C2 03 C2 3B 06 34 06 7F BD A1    0 |ﾇ ﾂ ﾂ; 4  ｽ。
0060 20 06 3C 02 B4 03 7C B4 B1 04 D2 E4 8A EC C7 45     < ｴ |ｴｱ ﾒ◢ ●ﾇE
0070 1A 00 00 89 55 1C 8A 47 08 3C FF 75 03 A0 23 06      U G < u  #
0080 88 05 C6 45 03 03 C7 45 20 FF FF C6 45 28 04 52     ﾆE  ﾇE   ﾆE( R
0090 B8 05 B5 F7 E2 89 55 0C 5A 8B DF B4 48 CD 18 80   ｸ ｵ  U Z■ﾟｴHﾍ _
00A0 47 03 05 80 67 03 07 B4 48 CD 18 80 47 03 05 80   G _g  ｴHﾍ _G _
00B0 67 03 07 01 57 08 80 F9 03 75 02 F7 DA 01 57 0A   g  W _  u ﾚ W
00C0 FE C9 75 D7 32 E4 CF                               ﾉuﾗ2◢ﾏ
```

5．0．2　拡大ＰＵＴ（漢字）

○概要

ＧＤＣの拡大文字描画機能を利用するものです．ＧＤＣにはＺＯＯＭコマンドがあって拡大描画係数を設定することによって１～１６倍のグラフィック文字の拡大描画が可能です．この機能を生かして，簡単に使えるようにＢＡＳＩＣコマンドに組み込む方法はないかと考えたのがこのプログラムで，ＰＵＴ文（漢字）に，拡大機能を追加しようというものです．画面に大きな文字でメニューを表示したいときなど便利に使えるでしょう．

○仕様

既存のＰＵＴ文と混乱を生じないよう，ＰＵＴの次に〝!〟をつけるものとします．書式は次のようになります．

PUT! (Sx, Sy), KANJI(〈コード〉)，〈拡大係数〉，〈パレット番号〉

〈拡大係数〉は１～１６の値をもつ数式で，省略時は１となります．

○使用方法

本プログラムは機械語でできています．ダンプリストまたは，ソースリストをモニタで打ち込んで，ＢＳＡＶＥでフロッピディスクにセーブしておきます．本プログラムを使用するときは，ＣＬＥＡＲ文，ＤＥＦ　ＳＥＧ文で機械語領域を定義してから，ＢＬＯＡＤ〝本プログラムの名前〟，Ｒとしてください．これ以後ＰＵＴ！文が使えるようになります．ＣＬＥＡＲ文，ＤＥＦ　ＳＥＧ文，ＢＬＯＡＤ文は，プログラム中でも，ダイレクトモードでもどちらでもＯＫです．ＢＬＯＡＤ実行後は，ＤＥＦ　ＳＥＧ文で定義を変更してもかまいません．高速ＣＩＲＣＬＥといっしょに使うときは，ＤＥＦ　ＳＥＧの値を調整して２つのプログラムが重ならないようにしてください．

〈例〉
```
CLEAR,&H1E00
DEF SEG=&H1E00
BLOAD"circle.bin",R
      本プログラムにつけた名前
  :
  :
PUT!(100,100),KANJI(&H3131),5
```

○プログラム説明

本プログラムは，高速ＣＩＲＣＬＥと異なり，ＢＡＳＩＣインタプリタ部――ＰＵＴ！文処理部を新しくつけ加えています．ＢＡＳＩＣの新しいコマンドを作ろうとするときには，拡張命令処理のエントリアドレスおよび拡張命令フラグを所定のエリアにセットすることで実現できます．本プログラムは，２つのプログラムから成り，１つは，この拡張命令処理アドレス等をセットするプログラムで，もう一つは，拡張命令の解釈および実行部です．ＢＬＯＡＤ"exput. bin"，Ｒを実行することにより，前者のプログラムが走り拡張命令が使えるようになるわけです．

〈N88-ＢＡＳＩＣの拡張命令機能〉

ＢＡＳＩＣインタプリタは，すべてのコマンドの解釈に先立ち，拡張命令フラグ(００６０Ｈ：１５９３Ｈ)が０以外の場合，(００６０Ｈ：１５９Ｃ～９ＤＨ)で指されるアドレスに制御を移します．拡張命令処理プログラム（ユーザ作成）は，そこで，コマンドの解釈を行ない拡張命令かどうかを調べます．もし，拡張命令なら，続いてそのコマンドの実行を行ない，キャリーフラグをセットしてリターンします．もし拡張命令でないなら，そのままキャリーフラグをクリアしてリターンします．キャリーフラグがクリアなら，インタプリタはコマンドの解釈を開始するのです．

〈アルゴリズム〉

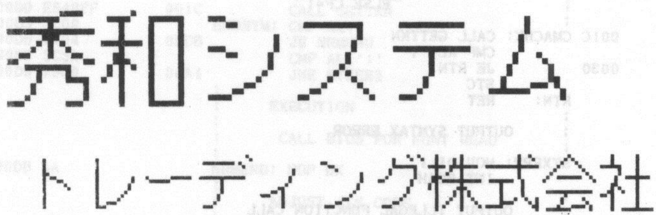
本ルーチンの処理は次のようになります．

①ＰＵＴコマンドかどうかを調べる．そうでないならばリターン．
②次のトークンが"！"かどうかを調べ，そうでないならばリターン．
③コマンドの解釈を行なって，座標，コード，拡大係数，パレット番号を得る．
④ＧＤＣのＺＯＯＭコマンドで拡大係数をセットする．
⑤ＢＩＯＳ（ＡＨ←００Ｈ，ＩＮＴ　１８Ｈ）でコードのフォントパターンを得る．
⑥ＢＩＯＳ　グラフィック文字の描画を用いて描画する．ただし，一度には，８×８のパターンしか描画できないので漢字の場合は，４回，半角は２回，４分の１角は１回，座標を変えながら描画を行う．

○サンプル

```
100 '拡大PUTデモンストレーション
110 CLEAR ,&H1E00
120 DEF SEG=&H1E00
130 BLOAD"exput.bin",R
140 '
150 SCREEN 3,1
160 A$="秀和システム"
170 ZOOM=5:X=10:Y=10
180 GOSUB *EXPUT
190 A$="トレーディング株式会社"
200 ZOOM=3:X=30:Y=130
210 GOSUB *EXPUT
220 END
230 *EXPUT
240 FOR I=3 TO LEN(A$)-2 STEP 2
250    CODE=ASC(MID$(A$,I))*256+ASC(MID$(A$,I+1))
260    PUT!(X,Y),KANJI(CODE),ZOOM,6
270    X=X+16*ZOOM
280 NEXT I
290 RETURN
```

○実行結果



○ソースリスト

```
                        ;
                        ;                EXPAND PUT KANJI SUBROUTINE
                        ;
      0620              SCRMOD   EQU WORD PTR .620H
      0623              FORCLR   EQU BYTE PTR .623H
      062E              VIEWX1   EQU WORD PTR .62EH
      0630              VIEWY1   EQU WORD PTR .630H
                        ;
      0640              PRMTBL   EQU 640H
      0640              COLOR    EQU BYTE PTR .PRMTBL+0
      0643              DIREC    EQU BYTE PTR .PRMTBL+3
      0648              XADRS    EQU WORD PTR .PRMTBL+8
      064A              YADRS    EQU WORD PTR .PRMTBL+0AH
      064C              DOTS1    EQU WORD PTR .PRMTBL+0CH
      065E              DOTS2    EQU WORD PTR .PRMTBL+1EH
      0660              PATN     EQU PRMTBL+20H
                        ;
      06EA              TEXPTR   EQU WORD PTR .6EAH
      104C              FONT     EQU 104CH
      141A              FACC1    EQU WORD PTR .141AH
      150A              WORK     EQU WORD PTR .150AH
                        ;
      1593              EXCOMF   EQU BYTE PTR .1593H
      159C              EXCOMO   EQU WORD PTR .159CH
      159E              EXCOMS   EQU WORD PTR .159EH
      15A0              EXFUNO   EQU WORD PTR .15A0H
      15A2              EXFUNS   EQU WORD PTR .15A2H
                        ;
                        ;            INITIALIZE EXTENDED COMMAND TABLE
                        ;
0000 16                 INIT:    PUSH SS
0001 1F                          POP DS
0002 B83B00                      MOV AX,OFFSET START
0005 A39C15                      MOV EXCOMO,AX
0008 B83F00                      MOV AX,OFFSET RETFAD
000B A3A015                      MOV EXFUNO,AX
```

```
000E 0E                        PUSH CS
000F 58                        POP AX
0010 A39E15                    MOV EXCOMS,AX
0013 A3A215                    MOV EXFUNS,AX
0016 C6069315FF                MOV EXCOMF,0FFH
001B CF                        IRET
                      ;
                      :     SUBROUTINES
                      :
                      :        GET A TOKEN
                      ;
001C BF0D00           GETTKN: MOV DI,13
001F CDC4                     INT 0C4H
0021 C3                       RET
                      ;
                      :        EVALUATE EXPRESSION
                      ;
                      :        AND CONVER TO INTEGER
                      ;
0022 BF4A00           GETVAL: MOV DI,74
0025 CDC4                     INT 0C4H
0027 C3                       RET
                      ;
                      :       GET A TOKEN
                      ;       AND CHECK IT
                      ;       IF IT IS ',' THEN CF=0
                      ;                    ELSE CF=1
                      ;
0028 E8F1FF      001C CMACHK: CALL GETTKN
002B 3C2C                     CMP AL,','
002D 7401        0030         JE RTN
002F F9                       STC
0030 C3               RTN:    RET
                      ;
                      ;       OUTPUT SYNTAX ERROR
                      ;
0031 BF0100           STXERR: MOV DI,1
0034 CDC4                     INT 0C4H
                      ;
                      ;       OUTPUT ILLEGAL FUNCTION CALL
                      ;
0036 BF0200           ILGLFN: MOV DI,2
0039 CDC4                     INT 0C4H
                      ;
                      ;     EXPANDED PUT MAIN ROUTINE
                      ;
                      ;       SYNTAX ANALYSIS
                      ;
003B 3CC1             START:  CMP AL,0C1H            ;PUT INTERNAL CODE
003D 7402        0041         JE PUTCOM              ;YES
003F F8               RETFAD: CLC
0040 CB                       RETF
0041 803C21           PUTCOM: CMP BYTE PTR [SI],'!'
0044 75F9        003F         JNE RETFAD
0046 A02306                   MOV AL,FORCLR
0049 A24006                   MOV COLOR,AL
004C C7060A150000             MOV WORK,0
0052 46                       INC SI
0053 8936EA06                 MOV TEXPTR,SI
0057 E8C2FF      001C         CALL GETTKN
005A 3C28                     CMP AL,'('
005C 75D3        0031         JNE STXERR
005E E8C1FF      0022         CALL GETVAL
0061 A11A14                   MOV AX,FACC1
0064 03062E06                 ADD AX,VIEWX1
0068 A34806                   MOV XADRS,AX
006B E8BAFF      0028         CALL CMACHK
006E 72C1        0031         JB STXERR
0070 E8AFFF      0022         CALL GETVAL
0073 A11A14                   MOV AX,FACC1
0076 03063006                 ADD AX,VIEWY1
007A A34A06                   MOV YADRS,AX
007D E89CFF      001C         CALL GETTKN
0080 3C29                     CMP AL,')'
0082 75AD        0031         JNE STXERR
0084 E8A1FF      0028         CALL CMACHK
0087 72A8        0031         JB STXERR
0089 E890FF      001C         CALL GETTKN
008C 80FBAB                   CMP BL,0ABH            ;KANJI INTERNAL CODE?
008F 75A0        0031         JNE STXERR
```

```
0091 E888FF     001C          CALL GETTKN
0094 3C28                     CMP AL,'('
0096 7599       0031          JNE STXERR
0098 E887FF     0022          CALL GETVAL
009B A11A14                   MOV AX,FACC1
009E 50                       PUSH AX
009F E87AFF     001C          CALL GETTKN
00A2 3C29                     CMP AL,')'
00A4 758B       0031 STXER2:  JNE STXERR
00A6 E87FFF     0028          CALL CMACHK
00A9 7228       00D3          JB ENDSTM
00AB E874FF     0022          CALL GETVAL
00AE A11A14                   MOV AX,FACC1
00B1 48                       DEC AX
00B2 A9F0FF                   TEST AX,0FFF0H
00B5 7403       00BA          JE NOTILG
00B7 E97CFF     0036 ILGLF2:  JMP ILGLFN
00BA A30A15          NOTILG:  MOV WORK,AX
00BD E868FF     0028          CALL CMACHK
00C0 7211       00D3          JB ENDSTM
00C2 E85DFF     0022          CALL GETVAL
00C5 A11A14                   MOV AX,FACC1
00C8 A9F8FF                   TEST AX,0FFF8H
00CB 75EA       00B7          JNE ILGLF2
00CD A24006                   MOV COLOR,AL
00D0 E849FF     001C          CALL GETTKN
00D3 3C00            ENDSTM:  CMP AL,0
00D5 7404       00DB          JE NRMEND
00D7 3C3A                     CMP AL,':'
00D9 75C9       00A4          JNE STXER2
                     ;
                     ;        EXECUTION
                     ;
                     ;         CALL BIOS FOR FONT READ
                     ;
00DB 5A              NRMEND:  POP DX
                     ;
                     ;        ADJUST JIS CODE
                     ;
00DC 80FE01                   CMP DH,1
00DF 7F10       00F1          JG KANJI                 ;ZENKAKU
00E1 B600                     MOV DH,0
00E3 740C       00F1          JE KANJI                 ;1/4KAKU
00E5 80C629                   ADD DH,29H               ;HANKAKU
00E8 80FA80                   CMP DL,80H
00EB 7204       00F1          JB KANJI
00ED 81C28000                 ADD DX,80H
00F1 8CDB            KANJI:   MOV BX,DS
00F3 B94C10                   MOV CX,FONT
00F6 B414                     MOV AH,14H
00F8 CD18                     INT 18H
                     ;
                     ;        SET ZOOM PARAMETER
                     ;
00FA E4A0            WAIT1:   IN AL,0A0H
00FC 2402                     AND AL,2
00FE 75FA       00FA          JNE WAIT1
0100 B046                     MOV AL,46H
0102 E6A2                     OUT 0A2H,AL
0104 D50A                     AAD
0106 E4A0            WAIT2:   IN AL,0A0H
0108 2402                     AND AL,2
010A 75FA       0106          JNE WAIT2
010C A10A15                   MOV AX,WORK
010F E6A0                     OUT 0A0H,AL
0111 8BE8                     MOV BP,AX
0113 45                       INC BP
0114 B103                     MOV CL,3
0116 D3E5                     SHL BP,CL
                     ;
                     ;       PREPARE PARAMETERS FOR BIOS
                     ;          GRAPHIC PUT
                     ;
0118 A12006                   MOV AX,SCRMOD
011B 3C02                     CMP AL,2
011D 7C0B       012A          JL LT2
011F 7404       0125          JE EQ2
0121 B5B0                     MOV CH,0B0H              ;SCRMOD=3
0123 EB1E       0143          JMPS NEXT
```

```
                                ;
0125 80CC08                     EQ2:    OR AH,08H               ;SCRMOD=2
0128 EB13               013D            JMPS COM
                                ;
012A 3C01                       LT2:    CMP AL,1
012C 7509               0137            JNE EQ0
012E 80FC03                             CMP AH,3                ;SCRMOD=1
0131 720A               013D            JB COM
0133 FEC4                               INC AH
0135 EB06               013D            JMPS COM
                                ;
0137 80C403                     EQ0:    ADD AH,3                ;SCRMOD=0
013A 80CC03                             OR AH,3
013D B104                       COM:    MOV CL,4
CP/M ASM86 1.1  SOURCE: EXPUT.A86

013F D2E4                               SHL AH,CL
0141 8AEC                               MOV CH,AH
0143 8B164C10                   NEXT:   MOV DX,WORD PTR .FONT
0147 8ACE                               MOV CL,DH
0149 BE4006                             MOV SI,PRMTBL
014C C606430602                         MOV DIREC,2
0151 012E4A06                           ADD YADRS,BP
0155 33C0                               XOR AX,AX
0157 A34C06                             MOV DOTS1,AX
015A A35E06                             MOV DOTS2,AX
015D BB0200                             MOV BX,2
0160 33FF                       LOOP1:  XOR DI,DI
0162 8A874C10                   LOOP2:  MOV AL,FONT[BX]
0166 88856006                           MOV PATN[DI],AL
016A 02DE                               ADD BL,DH
016C 47                                 INC DI
016D 83FF08                             CMP DI,8
0170 75F0               0162            JNE LOOP2
0172 B449                               MOV AH,49H
0174 87DE                               XCHG BX,SI
0176 CD18                               INT 18H
0178 87DE                               XCHG BX,SI
017A FEC9                               DEC CL
017C 7409               0187            JE SKIP1
017E 012E4806                           ADD XADRS,BP
0182 80EB0F                             SUB BL,0FH
0185 EBD9               0160            JMPS LOOP1
0187 012E4A06                   SKIP1:  ADD YADRS,BP
018B 2ADE                               SUB BL,DH
018D 43                                 INC BX
018E 80FE02                             CMP DH,2
0191 7504               0197            JNE SKIP2
0193 292E4806                           SUB XADRS,BP
0197 8ACE                       SKIP2:  MOV CL,DH
0199 FECA                               DEC DL
019B 75C3               0160            JNE LOOP1
019D F9                                 STC
019E CB                                 RETF
                                ;
                                        END

END OF ASSEMBLY.  NUMBER OF ERRORS:   0.  USE FACTOR:   1%
```

○ダンプリスト

```
0000 16 1F B8 3B 00 A3 9C 15 B8 3F 00 A3 A0 15 0E 58    ク; 」╭ ク? 」    X
0010 A3 9E 15 A3 A2 15 C6 06 93 15 FF CF BF 0D 00 CD   」、 」「 ニ ト  マソ  ヘ
0020 C4 C3 BF 4A 00 CD C4 C3 E8 F1 FF 3C 2C 74 01 F9   トテソJ ヘトテ♠円 く,t
0030 C3 BF 01 00 CD C4 BF 02 00 CD C4 3C C1 74 02 F8   テソ  ヘトソ  ヘトくチt
0040 CB 80 3C 21 75 F9 A0 23 06 A2 40 06 C7 06 0A 15   ヒ_く!u  # 「@ ヌ
0050 00 00 46 89 36 EA 06 E8 C2 FF 3C 28 75 D3 E8 C1     F 6◆ ♠ッ く(uモ♠チ
0060 FF A1 1A 14 03 06 2E 06 A3 48 06 E8 BA FF 72 C1    。    . 」H ♠コ rチ
0070 E8 AF FF A1 1A 14 03 06 30 06 A3 4A 06 E8 9C FF   ♠ッ 。     0 」J ♠╭
0080 3C 29 75 AD E8 A1 FF 72 A8 E8 90 FF 80 FB AB 75   く)uュ♠。 rィ♠┴ _ ォu
0090 A0 E8 88 FF 3C 28 75 99 E8 87 FF A1 1A 14 50 E8    ♠ く(u┐♠■ 。  P♠
00A0 7A FF 3C 29 75 8B E8 7F FF 72 28 E8 74 FF A1 1A   z く)u■♠  r(♠t 。
00B0 14 48 A9 F0 FF 74 03 E9 7C FF A3 0A 15 E8 68 FF    Hゥ× t ♥| 」 ♠h
00C0 72 11 E8 5D FF A1 1A 14 A9 F8 FF 75 EA A2 40 06   r ♠] 。 ゥ  u◆「@
00D0 E8 49 FF 3C 00 74 04 3C 3A 75 C9 5A 80 FE 01 7F   ♠I く t く:uノZ_
00E0 10 B6 00 74 0C 80 C6 29 80 FA 80 72 04 81 C2 80    カ t _ニ)_ _r ━ッ_
00F0 00 8C DB B9 4C 10 B4 14 CD 18 E4 A0 24 02 75 FA    ▮ロケL ェ ヘ ◢ $ u
0100 B0 46 E6 A2 D5 0A E4 A0 24 02 75 FA A1 0A 15 E6   -F◥「ュ ◢ $ u 。 ◥
0110 A0 8B E8 45 B1 03 D3 E5 A1 20 06 3C 02 7C 0B 74    ▮♠Eア モ◣。 く ¦ t
0120 04 B5 B0 EB 1E 80 CC 08 EB 13 3C 01 75 09 80 FC    オー♣ _フ ♣ く u _
0130 03 72 0A FE C4 EB 06 80 C4 03 80 CC 03 B1 04 D2    r  ト♣ _ト _フ ア メ
0140 E4 8A EC 8B 16 4C 10 8A CE BE 40 06 C6 06 43 06   ◢●▮  L ▮ホセ@ ニ C
0150 02 01 2E 4A 06 33 C0 A3 4C 06 A3 5E 06 BB 02 00     .J 3タ」L 」^ サ
0160 33 FF 8A 87 4C 10 88 85 60 06 02 DE 47 83 FF 08   3 ▮■L |▬`  ゛G▬
0170 75 F0 B4 49 87 DE CD 18 87 DE FE C9 74 09 01 2E   u×ェI■゛ヘ ■゛ ノt
0180 48 06 80 EB 0F EB D9 01 2E 4A 06 2A DE 43 80 FE   H _♣ ♣ル .J *゛C_
0190 02 75 04 29 2E 48 06 8A CE FE CA 75 C3 F9 CB        u ).H ▮ホ ハuテ ヒ
```

5．1　3次元グラフィックパッケージ

ここで紹介する3次元グラフィックパッケージは，N_{88}-BASICを拡張して，3次元命令をつけ加えたもので，3次元直線描画命令LINE＠文および，3次元視線設定命令VIEW＠文から成ります．VIEW＠文では，視点（眼の立置），注目点（注目している点）および，視野の広さを与え，LINE＠文で3次元空間上の2点を与えることによって，3次元空間に描かれた直線の視点から注目点を見た場合の像を画面に表示するもので，3次元描画を非常に簡単に実現できます．

○仕様

次の3つのステートメントをN_{88}-BASICから使えます．

1．　LINE＠［(W_{x1}, W_{y1}, W_{z1})］－ (W_{x2}, W_{y2}, W_{z2})［,〈パレット番号〉］

仮想的な3次元座標上で2点間にパレット番号で示された色で線を引きます．3次元座標系から画面への投影は，VIEW＠で設定した視線に従います．

2．　VIEW＠［(W_{X1}, W_{Y1}, W_{Z1})］［－(W_{X2}, W_{y2}, W_{z2})］［,〈視野の広さ〉］

3次元座標上にLINE＠で描かれた物体を見る視線を与えます．(W_{X1}, W_{Y1}, W_{Z1}) で視点（眼の位置）を与え，(W_{X2}, W_{Y2}, W_{Z2}) で注目点（注目している点）を与えます．〈視野の広さ〉は，画面の横方向の視野をラジアンで指定します．

○使用方法

本プログラムは機械語でできています．ダンプリストまたはソースリストをモニタで打ち込んで，BSAVEでフロッピディスクにしまっておきます．本プログラムを使用するときは，CLEAR文，DEF　SEG文で，機械語領域を定義してから，BLOAD〝本プログラムの名前〃，Rとしてください．これ以後，3次元命令が使えます．CLEAR文，DEF　SEG文，BLOAD文は，プログラム中でもダイレクトモードでもどちらでもOKです．BLOAD実行後はDEF　SEG文定義を変更してもかまいません．高速CIRCLE，拡大PU

Tとともに使われる場合は，ロードされたプログラムが重ならないように，DEF SEG文の値を調整してください．

〈例〉 CLEAR,&H1D00
　　DEF SEG=&H1D00
　　BLOAD"three.bin",R
　　　　　↑本プログラムにつけた名前
　　　　　・
　　　　　・

　　VIEW@(10, 50, 100)-(100, 200, 200), 1
　　LINE@(100, 100, 50)-(400, 500, 600)

○プログラムの説明

本プログラムは5．0の拡大PUTと同じ方法によってBASICコマンドを拡張しています．3次元ー2次元変換の方法は5．3の説明を参照してください．

○ソースリスト

```
                          ;
                          ;                THREE DIMENSION PACKAGE
                          ;
                          ;                 LINE@(X1,Y1,Z1)-(X2,Y2,Z2),COLOR
                          ;                 VIEW@(EX,EY,EZ)-(AX,AY,AZ),SIYA,KATAMUKI
                          ;
   0620                   SCRMOD  EQU WORD PTR .620H
   0623                   FORCLR  EQU BYTE PTR .623H
   062E                   VIEWX1  EQU WORD PTR .62EH
   0630                   VIEWY1  EQU WORD PTR .630H
   0632                   VIEWX2  EQU WORD PTR .632H
   0634                   VIEWY2  EQU WORD PTR .634H

   06EA                   TEXPTR  EQU WORD PTR .6EAH
   0A00                   ENTRY   EQU WORD PTR .0A00H
   104C                   WORK1   EQU WORD PTR .104CH
   104E                   WORK2   EQU WORD PTR .104EH
   1414                   BCVAL   EQU BYTE PTR .1414H
   141A                   FACC1   EQU 141AH
   1424                   FACC2   EQU 1424H
   150A                   LIOWRK  EQU 150AH
                          ;
   1593                   EXCOMF  EQU BYTE PTR .1593H
   159C                   EXCOMO  EQU WORD PTR .159CH
   159E                   EXCOMS  EQU WORD PTR .159EH
   15A0                   EXFUNO  EQU WORD PTR .15A0H
   15A2                   EXFUNS  EQU WORD PTR .15A2H
                          ;
                          ;              INITIALIZE EXTENDED COMMAND TABLE
                          ;
0000 16                   INIT:   PUSH SS
0001 1F                           POP DS
0002 B86D00                       MOV AX,OFFSET START
0005 A39C15                       MOV EXCOMO,AX
0008 B87900                       MOV AX,OFFSET RETFAD
```

```
000B A3A015                  MOV EXFUNO,AX
000E 0E                      PUSH CS
000F 58                      POP AX
0010 A39E15                  MOV EXCOMS,AX
0013 A3A215                  MOV EXFUNS,AX
0016 C6069315FF              MOV EXCOMF,0FFH
001B CF                      IRET
                    ;
                    ;    SUBROUTINES
                    ;
                    ;        GET A TOKEN
                    ;
001C BF0D00         GETTKN:  MOV DI,13
001F CDC4                    INT 0C4H
0021 C3                      RET
                    ;
                    ;       EVALUATE EXPRESSION
                    ;        AND CONVERT TO SINGLE
                    ;        AND STORE TO [BX-BX+3]
                    ;
0022 53             GETVAL:  PUSH BX
0023 F8                      CLC
0024 BF3A00                  MOV DI,58              ;EVAL
0027 CDC4                    INT 0C4H
0029 720B     0036           JB GVRET
002B BF2900                  MOV DI,41              ;REAL
002E CDC4                    INT 0C4H
0030 5B                      POP BX
0031 E84801   017C           CALL FAC1BX
0034 EBE6     001C           JMPS GETTKN
                    ;
0036 5B             GVRET:   POP BX
0037 EBE3     001C           JMPS GETTKN
                    ;
                    ;        EVALUATE '(X,Y,Z)'
                    ;        AND STORE TO [BX-BX+11]
                    ;
0039 53             GET3VL:  PUSH BX
003A E8E5FF   0022           CALL GETVAL
003D 5B                      POP BX
003E 3C2C                    CMP AL,','
0040 7521     0063           JNE STXERR
0042 83C304                  ADD BX,4
0045 53                      PUSH BX
0046 E8D9FF   0022           CALL GETVAL
0049 5B                      POP BX
004A 3C2C                    CMP AL,','
004C 7515     0063           JNE STXERR
004E 83C304                  ADD BX,4
0051 E8CEFF   0022           CALL GETVAL
0054 3C29                    CMP AL,')'
0056 750B     0063           JNE STXERR
0058 EBC2     001C           JMPS GETTKN
                    ;
                    ;       STATEMENT END CHECK
                    ;
005A 3C00           ENDCHK:  CMP AL,0
005C 7404     0062           JE ERET
005E 3C3A                    CMP AL,':'
0060 7501     0063           JNE STXERR
0062 C3             ERET:    RET
                    ;
                    ;       OUTPUT SYNTAX ERROR
                    ;
0063 BF0100         STXERR:  MOV DI,1
0066 CDC4                    INT 0C4H
                    ;
                    ;       OUTPUT ILLEGAL FUNCTION CALL
                    ;
0068 BF0200         ILGLFN:  MOV DI,2
006B CDC4                    INT 0C4H
                    ;
                    ;       COMMAND ANALYSIS
                    ;
006D 3CB0           START:   CMP AL,0B0H            ;LINE
006F 750A     007B           JNE NOTLIN
0071 803C40                  CMP BYTE PTR [SI],'@'
0074 7503     0079           JNE RETFAD
```

```
0076 E95D01        01D6            JMP LINE
0079 F8                   RETFAD:  CLC
007A CB                            RETF
007B 3CFF                 NOTLIN:  CMP AL,0FFH               ;VIEW
007D 75FA          0079            JNE RETFAD
007F 803C85                        CMP BYTE PTR [SI],85H
0082 75F5          0079            JNE RETFAD
0084 2EC606FC0300                  MOV CS:VIEWFG,0
008A 807C0140                      CMP BYTE PTR 1[SI],'@'
008E 75E9          0079            JNE RETFAD
0090 46                   VIEWA:   INC SI
0091 46                            INC SI
0092 8936EA06                      MOV TEXPTR,SI
0096 E883FF        001C            CALL GETTKN
0099 3C28                          CMP AL,'('
009B 7506          00A3            JNE HYPH
009D BB9403                        MOV BX,OFFSET VEX
00A0 E896FF        0039            CALL GET3VL
00A3 80FBF4               HYPH:    CMP BL,0F4H
00A6 750D          00B5            JNE SIYQ
00A8 E871FF        001C            CALL GETTKN
00AB 3C28                          CMP AL,'('
00AD 75B4          0063            JNE STXERR
00AF BBA003                        MOV BX,OFFSET VAX
00B2 E884FF        0039            CALL GET3VL
00B5 3C2C                 SIYQ:    CMP AL,','
00B7 7510          00C9            JNE STMEND
00B9 BBC403                        MOV BX,OFFSET SIYA
00BC E863FF        0022            CALL GETVAL
00BF 3C2C                          CMP AL,','
00C1 7506          00C9            JNE STMEND
00C3 BBC803                        MOV BX,OFFSET KATAMU
00C6 E859FF        0022            CALL GETVAL
00C9 E88EFF        005A   STMEND:  CALL ENDCHK
                          ;       EXECUTION VIEW
                          ;
                          ;        VDX=VAX-VEX
                          ;        VDX2=VDX*VDX
                          ;        VDY=VAY-VEY
                          ;        VDY2=VDY*VDY
                          ;        VDZ=VAZ-VEZ
                          ;        VDZ2=VDZ*VDZ
                          ;
00CC 33F6                          XOR SI,SI
00CE 56                   VLOP1:   PUSH SI
00CF 8D9CA003                      LEA BX,OFFSET VAX[SI]
00D3 E8C800        019E            CALL BXFAC2
00D6 8D9C9403                      LEA BX,OFFSET VEX[SI]
00DA E8B000        018D            CALL BXFAC1
00DD BF2C00                        MOV DI,44                 ;SUBTRACT
00E0 CDC4                          INT 0C4H
00E2 5E                            POP SI
00E3 56                            PUSH SI
00E4 8D9CAC03                      LEA BX,OFFSET VDX[SI]
00E8 E89100        017C            CALL FAC1BX
00EB E8B000        019E            CALL BXFAC2
00EE BF2D00                        MOV DI,45                 ;MULTIPLY
00F1 CDC4                          INT 0C4H
00F3 5E                            POP SI
00F4 8D9CB803                      LEA BX,OFFSET VDX2[SI]
00F8 E88100        017C            CALL FAC1BX
00FB 83C604                        ADD SI,4
00FE 83FE0C                        CMP SI,12
0101 75CB          00CE            JNE VLOP1
                          ;
                          ;        TEMP=SQRT(VDX2+VDY2)
                          ;        COSA=VDX/TEMP
                          ;        SINA=VDY/TEMP
                          ;        TEMP=SQRT(VDX2+VDZ2)
                          ;        COSB=VDX/TEMP
                          ;        SINB=VDY/TEMP
                          ;
0103 33F6                          XOR SI,SI
0105 56                   VLOP2:   PUSH SI
0106 8D9CBC03                      LEA BX,OFFSET VDY2[SI]
010A E88000        018D            CALL BXFAC1
010D BBB803                        MOV BX,OFFSET VDX2
0110 E88B00        019E            CALL BXFAC2
```

```
0113 BF2B00                        MOV DI,43               ;ADD
0116 CDC4                          INT 0C4H
0118 E86202         037D           CALL SQRT
011B E89100         01AF           CALL SAVE
011E BBAC03                        MOV BX,OFFSET VDX
0121 E87A00         019E           CALL BXFAC2
0124 BF2E00                        MOV DI,46               ;DIVIDE
0127 CDC4                          INT 0C4H
0129 5E                            POP SI
012A 8D9CEC03                      LEA BX,OFFSET COSA[SI]
012E E84B00         017C           CALL FAC1BX
0131 E88800         01BC           CALL LOAD
0134 56                            PUSH SI
0135 BBB003                        MOV BX,OFFSET VDY
0138 E86300         019E           CALL BXFAC2
013B BF2E00                        MOV DI,46               ;DIVIDE
013E CDC4                          INT 0C4H
0140 5E                            POP SI
0141 8D9CF403                      LEA BX,OFFSET SINA[SI]
0145 E83400         017C           CALL FAC1BX
0148 83C604                        ADD SI,4
014B 83FE08                        CMP SI,8
014E 75B5           0105           JNE VLOP2
                         ;
                         ;         R=1/TAN(SIYA/2)
                         ;
0150 BBC403                        MOV BX,OFFSET SIYA
0153 E83700         018D           CALL BXFAC1
0156 802E1D1401                    SUB BYTE PTR .FACC1+3,1 ;FACC1=FACC1/2
015B 7304           0161           JNB NOZERO
015D FE061D14                      INC BYTE PTR .FACC1+3
0161 E81F02         0383 NOZERO:   CALL TAN
0164 33C0                          XOR AX,AX
0166 A32414                        MOV WORD PTR .FACC2,AX
0169 B80081                        MOV AX,8100H            ;FACC2=1
016C A32614                        MOV WORD PTR .FACC2+2,AX
016F BF2E00                        MOV DI,46               ;DIVIDE
0172 CDC4                          INT 0C4H
0174 BBE403                        MOV BX,OFFSET R
0177 E80200         017C           CALL FAC1BX
017A F9                            STC
017B CB                            RETF
                         ;
                         ;     SUBROUTINES
                         ;
                         ;         [FACC1,FACC1+3]=>[BX,BX+3]
                         ;
017C A11A14              FAC1BX:   MOV AX,WORD PTR .FACC1
017F 2E89870000                    MOV CS:[BX],AX
0184 A11C14                        MOV AX,WORD PTR .FACC1+2
0187 2E89870200                    MOV CS:2[BX],AX
018C C3                            RET
                         ;
                         ;         [BX,BX+3]=>[FACC1,FACC1+3]
                         ;
018D 2E8B870000          BXFAC1:   MOV AX,CS:[BX]
0192 A31A14                        MOV WORD PTR .FACC1,AX
0195 2E8B870200                    MOV AX,CS:2[BX]
019A A31C14                        MOV WORD PTR .FACC1+2,AX
019D C3                            RET
                         ;
                         ;         [BX,BX+3]=>[FACC2,FACC2+3]
                         ;
019E 2E8B870000          BXFAC2:   MOV AX,CS:[BX]
01A3 A32414                        MOV WORD PTR .FACC2,AX
01A6 2E8B870200                    MOV AX,CS:2[BX]
01AB A32614                        MOV WORD PTR .FACC2+2,AX
01AE C3                            RET
                         ;
                         ;         [FACC1,FACC1+3]=>[WORK1,WORK1+3]
                         ;
01AF A11A14              SAVE:     MOV AX,WORD PTR .FACC1
01B2 A34C10                        MOV WORK1,AX
01B5 A11C14                        MOV AX,WORD PTR .FACC1+2
01B8 A34E10                        MOV WORK2,AX
01BB C3                            RET
                         ;
                         ;         [WORK1,WORK1+3]=>[FACC1,FACC1+3]
                         ;
```

```
01BC A14C10              LOAD:   MOV AX,WORK1
01BF A31A14                      MOV WORD PTR .FACC1,AX
01C2 A14E10                      MOV AX,WORK2
01C5 A31C14                      MOV WORD PTR .FACC1+2,AX
01C8 C3                          RET
                         ;
                         ;       [WORK1,WORK1+3]=>[FACC2,FACC2+3]
                         ;
01C9 A14C10              LOAD2:  MOV AX,WORK1
01CC A32414                      MOV WORD PTR .FACC2,AX
01CF A14E10                      MOV AX,WORK2
01D2 A32614                      MOV WORD PTR .FACC2+2,AX
01D5 C3                          RET
                         ;
                         ;      LINE@ SYNTAX ANALYSIS
                         ;
01D6 46                  LINE:   INC SI
01D7 8936EA06                    MOV TEXPTR,SI
01DB 2E803EFC0300                CMP CS:VIEWFG,0
01E1 752C           020F         JNE NVIEW
                         ;
                         ;       RR=(VIEWX2-VIEWX1)*R
                         ;
01E3 A13206                      MOV AX,VIEWX2
01E6 2B062E06                    SUB AX,VIEWX1
01EA D1F8                        SAR AX,1
01EC A31A14                      MOV WORD PTR .FACC1,AX
01EF C606141402                  MOV BCVAL,2
01F4 BF2900                      MOV DI,41              ;REAL
01F7 CDC4                        INT 0C4H
01F9 BBE403                      MOV BX,OFFSET R
01FC E89FFF         019E         CALL BXFAC2
01FF BF2D00                      MOV DI,45              ;MULTIPLY
0202 CDC4                        INT 0C4H
0204 BBE803                      MOV BX,OFFSET RR
0207 E872FF         017C         CALL FAC1BX
020A 2EFE06FC03                  INC CS:VIEWFG
020F E80AFE         001C NVIEW:  CALL GETTKN
0212 80FBF4                      CMP BL,0F4H
0215 7412           0229         JE OMIT
                         ;
                         ;      EVALUATE (X1,Y1,Z1)
                         ;
0217 3C28                        CMP AL,'('
0219 7403           021E         JE NOER
021B E945FE         0063 TOSTXE: JMP STXERR
021E BBCC03              NOER:   MOV BX,OFFSET LPX
0221 E815FE         0039         CALL GET3VL
0224 80FBF4                      CMP BL,0F4H
0227 75F2           021B         JNE TOSTXE
0229 BB0A15              OMIT:   MOV BX,LIOWRK
022C E84100         0270         CALL CONV
                         ;
                         ;EVALUATE (X2,Y2,Z2)
                         ;
022F E8EAFD         001C         CALL GETTKN
0232 3C28                        CMP AL,'('
0234 75E5           021B         JNE TOSTXE
0236 BBCC03                      MOV BX,OFFSET LPX
0239 E8FDFD         0039         CALL GET3VL
023C 50                          PUSH AX
023D BB0E15                      MOV BX,LIOWRK+4
0240 E82D00         0270         CALL CONV
0243 58                          POP AX
                         ;
                         ;EVALUATE COLOR OPERAND
                         ;
0244 3C2C                        CMP AL,','
0246 A02306                      MOV AL,FORCLR
0249 B400                        MOV AH,0
024B 7515           0262         JNE NOTCMA
024D BF4A00                      MOV DI,74              ;EVAL&INT
0250 CDC4                        INT 0C4H
0252 A11A14                      MOV AX,WORD PTR .FACC1
0255 A9F8FF                      TEST AX,0FFF8H
0258 7403           025D         JE LE7
025A E90BFE         0068 TOILG:  JMP ILGLFN
025D 50                  LE7:    PUSH AX
025E E8BBFD         001C         CALL GETTKN
```

```
0261 58                         POP AX
0262 BB0A15           NOTCMA:   MOV BX,LIOWRK
0265 894708                     MOV 8[BX],AX
0268 C6470A00                   MOV BYTE PTR 10[BX],0
026C CDA7                       INT 0A7H              ;LIO GLINE
026E F9                         STC
026F CB                         RETF
                      ;
                      ;         CONVERT FROM 3D TO 2D
                      ;
                      ;         XX=LPX-VEX
                      ;         YY=LPY-VEY
                      ;         ZZ=LPZ-VEZ
                      ;
0270 53               CONV:     PUSH BX
0271 33F6                       XOR SI,SI
0273 56               CLOP1:    PUSH SI
0274 8D9CCC03                   LEA BX,OFFSET LPX[SI]
0278 E823FF      019E           CALL BXFAC2
027B 8D9C9403                   LEA BX,OFFSET VEX[SI]
027F E80BFF      018D           CALL BXFAC1
0282 BF2C00                     MOV DI,44             ;SUBTRACT
0285 CDC4                       INT 0C4H
0287 5E                         POP SI
0288 8D9CD803                   LEA BX,OFFSET XX[SI]
028C E8EDFE      017C           CALL FAC1BX
028F 83C604                     ADD SI,4
0292 83FE0C                     CMP SI,12
0295 75DC        0273           JNE CLOP1
                      ;
                      ;         TEMP=XX*COSA-YY*SINA
                      ;         YY=YY*COSA+XX*SINA
                      ;         XX=TEMP
                      ;         TEMP=XX*COSB-ZZ*SINB
                      ;         ZZ=ZZ*COSB+XX*SINB
                      ;         XX=TEMP
                      ;
0297 33F6                       XOR SI,SI
0299 56               CLOP2:    PUSH SI
029A BBD803                     MOV BX,OFFSET XX
029D E8EDFE      018D           CALL BXFAC1
02A0 8D9CEC03                   LEA BX,OFFSET COSA[SI]
02A4 E8F7FE      019E           CALL BXFAC2
02A7 BF2D00                     MOV DI,45             ;MULTIPLY
02AA CDC4                       INT 0C4H
02AC E800FF      01AF           CALL SAVE
02AF 5E                         POP SI
02B0 56                         PUSH SI
02B1 8D9CDC03                   LEA BX,OFFSET YY[SI]
02B5 E8D5FE      018D           CALL BXFAC1
02B8 8D9CF403                   LEA BX,OFFSET SINA[SI]
02BC E8DFFE      019E           CALL BXFAC2
02BF BF2D00                     MOV DI,45             ;MULTIPLY
02C2 CDC4                       INT 0C4H
02C4 E802FF      01C9           CALL LOAD2
02C7 BF2C00                     MOV DI,44             ;SUBTRACT
02CA CDC4                       INT 0C4H
02CC BBD803                     MOV BX,OFFSET XX
02CF E8CCFE      019E           CALL BXFAC2
02D2 E8A7FE      017C           CALL FAC1BX
02D5 5E                         POP SI
02D6 56                         PUSH SI
02D7 8D9CF403                   LEA BX,OFFSET SINA[SI]
02DB E8AFFE      018D           CALL BXFAC1
02DE BF2D00                     MOV DI,45             ;MULTIPLY
02E1 CDC4                       INT 0C4H
02E3 E8C9FE      01AF           CALL SAVE
02E6 5E                         POP SI
02E7 56                         PUSH SI
02E8 8D9CDC03                   LEA BX,OFFSET YY[SI]
02EC E89EFE      018D           CALL BXFAC1
02EF 8D9CEC03                   LEA BX,OFFSET COSA[SI]
02F3 E8A8FE      019E           CALL BXFAC2
02F6 BF2D00                     MOV DI,45             ;MULTIPLY
02F9 CDC4                       INT 0C4H
02FB E8CBFE      01C9           CALL LOAD2
02FE BF2B00                     MOV DI,43             ;ADD
0301 CDC4                       INT 0C4H
```

```
0303 5E                         POP SI
0304 8D9CDC03                   LEA BX,OFFSET YY[SI]
0308 E871FE         017C        CALL FAC1BX
030B 83C604                     ADD SI,4
030E 83FE08                     CMP SI,8
0311 7586           0299        JNE CLOP2
                    ;
                    ;           [BX+2,BX+3]=(VIEWY2-VIEWY1)/2-ZZ/XX*RR
                    ;
0313 BBD803                     MOV BX,OFFSET XX
0316 E874FE         018D        CALL BXFAC1
0319 BBE003                     MOV BX,OFFSET ZZ
031C E87FFE         019E        CALL BXFAC2
031F BF2E00                     MOV DI,46                  ;DIVIDE
0322 CDC4                       INT 0C4H
0324 BBE803                     MOV BX,OFFSET RR
0327 E874FE         019E        CALL BXFAC2
032A BF2D00                     MOV DI,45                  ;MULTIPLY
032D CDC4                       INT 0C4H
032F BF0C00                     MOV DI,12                  ;INT
0332 CDC4                       INT 0C4H
0334 A13406                     MOV AX,VIEWY2
0337 2B063006                   SUB AX,VIEWY1
033B D1F8                       SAR AX,1
033D 2B061A14                   SUB AX,WORD PTR .FACC1
0341 5B                         POP BX
0342 894702                     MOV 2[BX],AX
                    ;
                    ;           [BX,BX+1]=YY/XX*R+(VIEWX2-VIEWX1)/2
                    ;
0345 53                         PUSH BX
0346 BBD803                     MOV BX,OFFSET XX
0349 E841FE         018D        CALL BXFAC1
034C BBDC03                     MOV BX,OFFSET YY
034F E84CFE         019E        CALL BXFAC2
0352 C606141404                 MOV BCVAL,4
0357 BF2E00                     MOV DI,46                  ;DIVIDE
035A CDC4                       INT 0C4H
035C BBE803                     MOV BX,OFFSET RR
035F E83CFE         019E        CALL BXFAC2
0362 BF2D00                     MOV DI,45                  ;MULTIPLY
0365 CDC4                       INT 0C4H
0367 BF0C00                     MOV DI,12                  ;INT
036A CDC4                       INT 0C4H
036C A13206                     MOV AX,VIEWX2
036F 2B062E06                   SUB AX,VIEWX1
0373 D1F8                       SAR AX,1
0375 03061A14                   ADD AX,WORD PTR .FACC1
0379 5B                         POP BX
037A 8907                       MOV [BX],AX
037C C3                         RET
                    ;
                    ;           SQRT & TAN ROUTINE
                    ;
                    ;           ENTRY MUST BE ADJUSTED
                    ;
037D 2EA19003       SQRT:       MOV AX,CS:SQRTAD
0381 EB04           0387        JMPS CALFUN
0383 2EA19203       TAN:        MOV AX,CS:TANAD
                    ;
0387 A3000A         CALFUN:     MOV ENTRY,AX
038A BF1500                     MOV DI,21                  ;SYS CALL
038D CDC4                       INT 0C4H
038F C3                         RET
                    ;
                    ;
                    ;           WORK AREA
                    ;
                    ;
0390 7C4F           SQRTAD      DW 4F7CH
0392 F54B           TANAD       DW 4BF5H
                    ;
0394 00000000       VEX         DW 0,0
0398 00000000       VEY         DW 0,0
039C 00000000       VEZ         DW 0,0
03A0 00000081       VAX         DW 0,8100H
03A4 00000000       VAY         DW 0,0
03A8 00000000       VAZ         DW 0,0
03AC 00000000       VDX         DW 0,0
```

```
03B0 00000000          VDY      DW 0,0
03B4 00000000          VDZ      DW 0,0
03B8 00000000          VDX2     DW 0,0
03BC 00000000          VDY2     DW 0,0
03C0 00000000          VDZ2     DW 0,0
03C4 00000081          SIYA     DW 0,8100H
03C8 00000000          KATAMU   DW 0,0
03CC 00000000          LPX      DW 0,0
03D0 00000000          LPY      DW 0,0
03D4 00000000          LPZ      DW 0,0
03D8 00000000          XX       DW 0,0
03DC 00000000          YY       DW 0,0
03E0 00000000          ZZ       DW 0,0
03E4 6C4D6A81          R        DW 4D6CH,816AH
03E8 00000000          RR       DW 0,0
03EC 00000081          COSA     DW 0,8100H
03F0 00000081          COSB     DW 0,8100H
03F4 00000000          SINA     DW 0,0
03F8 00000000          SINB     DW 0,0
03FC 00                VIEWFG   DB 0
                       ;
                                END
```

5．2　ポリゴン分割による三次元グラフィクス

5．2．1　概要

このプログラムは，物体表面をポリゴン（多角形）の集合であると考えて，分割され，座標値を計算されたポリコン群の透視図を，ディスプレイに表示するプログラムです．

入力するデータは，ポリゴンの座標値，頂点の法線ベクトル，ポリゴンの色，視点，視線上の一点，視野の角度です．出力はカラーディスプレイ上へのドット出力として行なわれます．

なお，入力データとして取り扱うポリゴンは，処理の関係上，三角形に限定されており，またシェイディングを行うため，ポリゴンの色は6色となります．このプログラムは，Romneyのアルゴリズム（参考文献参照）を参考にしてあります．処理はできるだけ単純化を図るように努力しました．

プログラムのおおまかな流れを以下に示します．

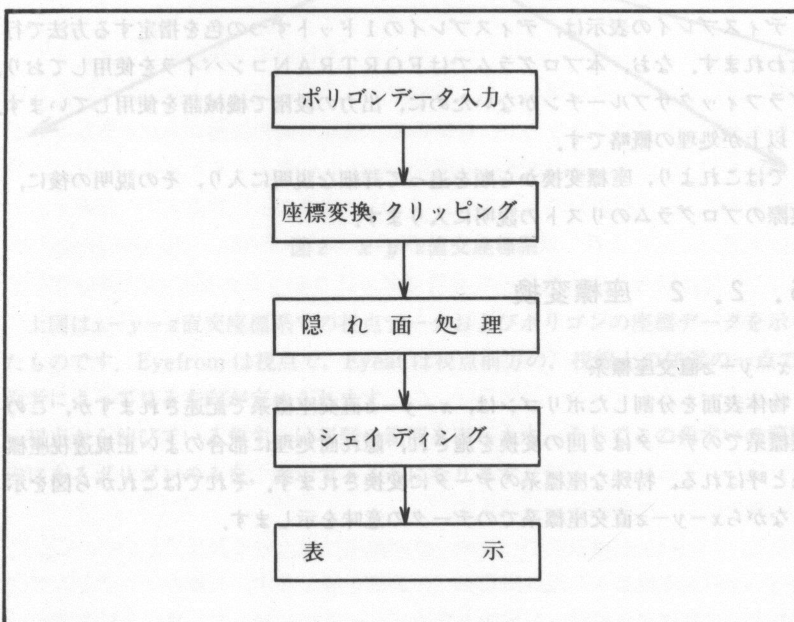


図1　処理の概要

ではこれより，図１の個々の内容の概略を説明します．

ポリゴンデータ入力は，ポリゴンの頂点の（x，y，z）値および，その点での法線ベクトル（Nx，Ny，Nz），ポリゴンの色を入力するものです．

座標変換は，一般のｘ－ｙ－ｚ直交座標系で記述 されたポリゴン頂点値のデータを，後の隠れ面処理に都合のよいように，正規透視座標系という特殊な座標系に変換するプロセスです．

隠れ面処理は，三次元図形処理で最も重要となる処理で，このプログラムにおいても，最も複雑なプロセスになっています．隠れ面処理の目的は，物体の後ろに隠れたものは表示しない，つまり，目に近い物体から優先的に表示することにあります．この処理を施すことによって，物体の前後関係のはっきりした透視図を描くことができます．

シェイディングは，光源を仮定した時の物体表面の陰影づけを行なうもので，ポリゴンデータの法線ベクトルと，光線のベクトルとの計算によって行なわれます．なお，ポリゴンの各頂点によって，陰影の度合いは異なるため，ポリゴン内部の点を表示する時には，陰影を補間し，円滑化するスムーズシェイディング処理が必要となります．

ディスプレイの表示は，ディスプレイの１ドットずつの色を指定する方法で行なわれます．なお，本プログラムではＦＯＲＴＲＡＮコンパイラを使用しており，グラフィックサブルーチンがないために，出力の段階で機械語を使用しています．

以上が処理の概略です．

ではこれより，座標変換から順を追って詳細な説明に入り，その説明の後に，実際のプログラムのリストの説明に入ります．

５．２．２　座標変換

○$x-y-z$直交座標系

物体表面を分割したポリゴンは，$x-y-z$直交座標系で記述されますが，この座標系でのデータは２回の変換を施され，隠れ面処理に都合のよい正規透視座標系と呼ばれる，特殊な座標系のデータに変換されます．それではこれから図を示しながら$x-y-z$直交座標系でのデータの意味を示します．

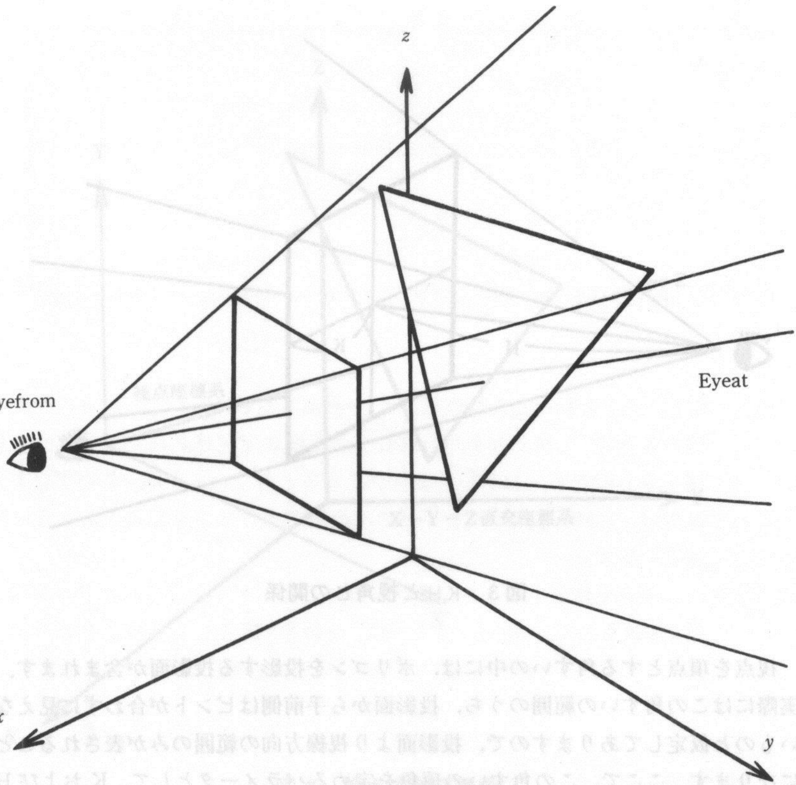


図2　x-y-z直交座標系

上図は$x-y-z$直交座標系での視点データおよびポリゴンの座標データを示したものです．Eyefromは視点で，Eyeatは視点前方の，視線上の任意の一点で，両者によって見る方向が定められます．

視点から伸びている角すいは視野の範囲を表します．そしてこの角すいの範囲内にあるポリゴンのみを，表示することになります．

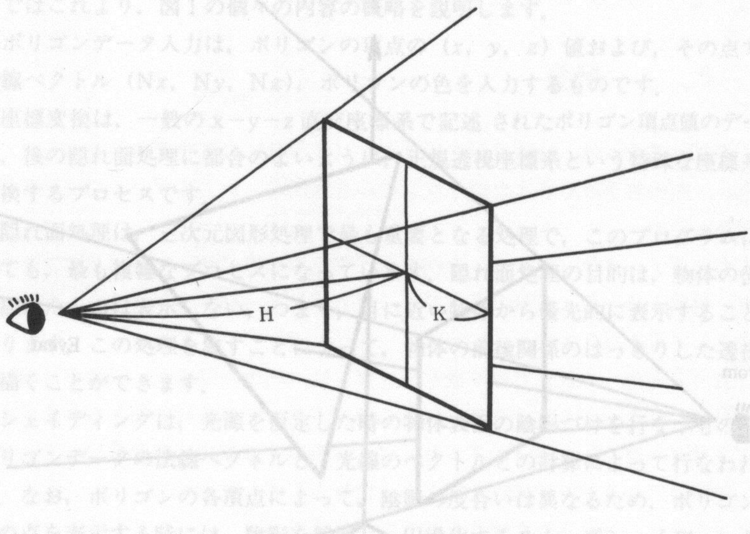


図3　K,H と視角との関係

視点を頂点とする角すいの中には，ポリゴンを投影する投影面が含まれます．実際にはこの角すいの範囲のうち，投影面から手前側はピントが合わずに見えないものと仮定してありますので，投影面より視線方向の範囲のみが表されることになります．ここで，この角すいの頂角を定めるパラメータとして，KおよびHを考えます．Kは投影面の大きさの1/2，Hは視点から投影面までの距離を示しています．これにより，頂角すなわち視角θは2×Arctan（K/H）となります．ここで，視角のみを問題にする時はK/Hという比の値だけで済むのですが，投影面と視点の間のポリゴンは見えないものとしてクリッピングしますので，Hという値が重要になってきます．

視角θが大きくなると，投影像は，カメラの広角レンズで撮った像のように歪んできますが，より多くの範囲を表示することが可能となります．またθが小さくなると，望遠レンズで撮ったように，奥行きがあまり感じられないが，より狭い範囲を拡大した投影像になります．

さて，$x-y-z$座標系でのパラメータの説明は以上のとおりでしたが，ここで$x-y-z$直交座標系から視点座標系への変換を施します．実際のプログラムでは，この次の正規透視座標系への変換も含めて，一度に行なってしまうわけですが，ここではわかり易くするために，二度の変換に分けて考えてみます．

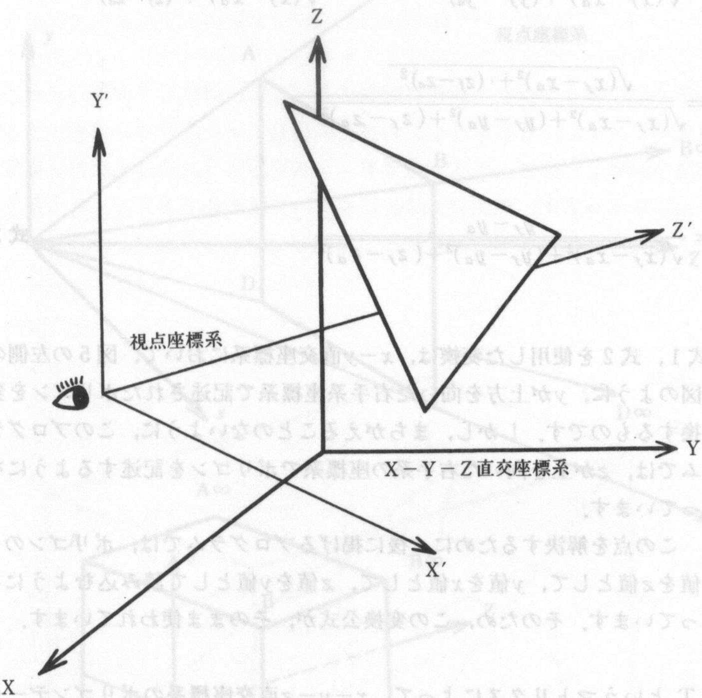


図4　視点透視座標系

○視点透視座標系

視点透視座標系は視点をz軸に，水平面と平行にx軸をとった座標系で，$x-y-z$直交座標系との関係は図4のようになります．

ここで$x-y-z$直交座標系から視点座標系への変換行列を式1に掲げます．

式1

$$Tv=\begin{pmatrix} \cos\alpha & \sin\alpha\sin\beta & \sin\alpha\cos\beta \\ 0 & \cos & -\sin\beta \\ \sin\alpha & -\cos\alpha\sin\beta & -\cos\alpha\cos\beta \end{pmatrix}$$

式中の　$\cos\alpha$，$\sin\alpha$，$\cos\beta$，$\sin\beta$は次の式によって計算されます．

$$\cos\alpha = \frac{z_f - z_a}{\sqrt{(x_f - x_a)^2 + (y_f - y_a)^2}} \quad \sin\alpha = \frac{x_a - x_f}{\sqrt{(x_f - x_a)^2 + (z_f - z_a)^2}}$$

$$\cos\beta = \frac{\sqrt{(x_f - x_a)^2 + (z_f - z_a)^2}}{\sqrt{(x_f - x_a)^2 + (y_f - y_a)^2 + (z_f - z_a)^2}}$$

$$\sin\beta = \frac{y_f - y_a}{\sqrt{(x_f - x_a)^2 + (y_f - y_a)^2 + (z_f - z_a)^2}} \qquad \textbf{式 2}$$

注）　式１，式２を使用した変換は，$x-y$直交座標系において，図５の左側の図のように，yが上方を向いた右手系座標系で記述されたポリゴンを変換するものです．しかし，まちがえることのないように，このプログラムでは，zが上を向いた右手系の座標系でポリゴンを記述するようになっています．

この点を解決するために，後に掲げるプログラムでは，ポリゴンのx値をz値として，y値をx値として，z値をy値として読み込むようになっています．そのため，この変換公式が，そのまま使われています．

式１のT_vというマトリクスによって，$x-y-z$直交座標系のポリゴンデータは，視点座標のデータへと変換されます．

○正規透視座標系

正規透視座標系は，隠面処理にたいへん便利な座標系であり，この座標系に変換されたポリゴンのデータを使用すると，容易に隠面消去処理を行なうことができます．正規透視座標系は図５に示した性質をもった座標系です．すなわち，視野内の角すいのうち，投影面から視る線方向に広がる領域を，直方体に押し縮めた形となります．ここで例を掲げますと，視点座標系の（0，0，∞）の点は，正視透視座標系での（0，0，1）の点に，また投影面上の点（0，0，H）は，（0，0，0）に変換されます．このように視点座標系から正規透視座標系への変換において，視線上の点はz値のみが変化しますが，それ以外の点では，x，y値も変化します．つまり，視点より遠い場所にあるポリゴンほど大きさが縮小されて，正規透視座標系の直方体の中に納められます．逆に視点近くのポリゴンは，拡大されることになります．

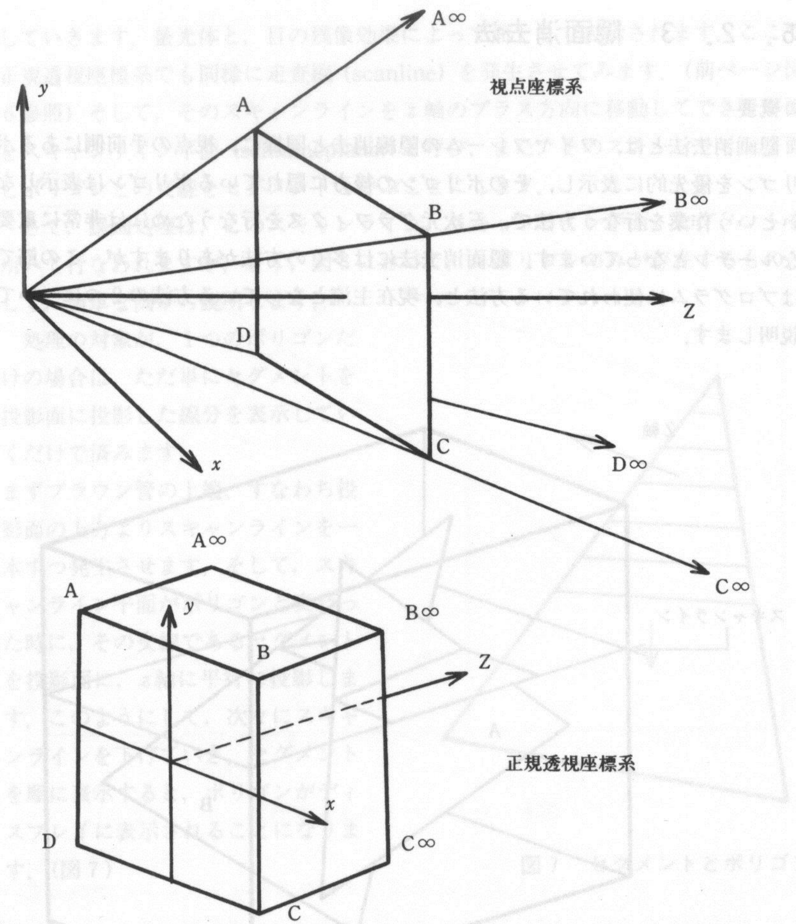


図5　正規透視座標系の性質

ここで視点座標系から正規透視座標系への変換マトリクスを書くべきですが，4×4マトリクスになるなど，話が難解になるおそれがあるので，ここでは控えます．

さて，以上の2つの変換によって$x-y-z$直交座標系で表わされたポリゴンデータは，隠面処理に便利な正視透視座標系へと変換されました．では，これから，この変換されたデータをどのように使って隠面処理を行なうかについて，説明していきます．

5．2．3　隠面消去法

○概要

隠面消去法とは，ワイヤフレームの隠線消去と同様に，視点の手前側にあるポリゴンを優先的に表示し，そのポリゴンの後方に隠れているポリゴンは表示しないという作業を行なう方法で，三次元グラフィクスを行なうためには非常に重要なルーチンとなっています．隠面消去法には多くの方法がありますが，この節ではプログラムに使われている方法と，現在主流となっている方法の2つについて説明します．

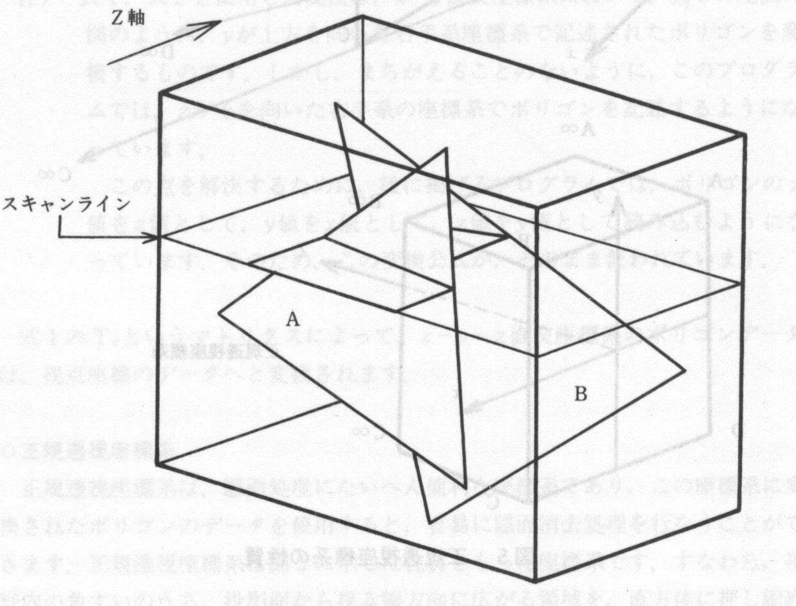


図6　正規透視座標系とスキャンライン

まず最初にこのプログラムで使われている方法を説明します．正視透視座標系に変換されたポリゴンは，拡大，縮小を受けて，正規透視座標系の直方体の中に納められます．ここで，この直方体をテレビとして考えてみます．すると投影面がテレビのブラウン管表面に対応します．テレビの原理は，上から下へ走査線が走り，その走査線上を電子ビームが左から右へ走って，1ドットごとに点を表示

していきます．螢光体と，目の残像効果によって面として表示されます．ここで，正規透視座標系でも同様に走査線（scanline）を発生させてみます．（前ページ図6参照）そして，そのスキャンラインを z 軸のプラス方向に移動してできた平面をスキャンライン平面（scanlineplane）と呼び，また，そのスキャンライン平面とポリゴンとの交線をセグメントと呼ぶことにします．

さて，隠画処理は，このスキャンライン，スキャンライン平面，セグメントを用いて行なわれますが，まず，図6において，Aポリゴンのみが存在するものとして，簡単な例から説明します．

処理の対象が，１つのポリゴンだけの場合は，ただ単にセグメントを投影面に投影した線分を表示していくだけで済みます．
まずブラウン管の上端、すなわち投影面の上方よりスキャンラインを一本ずつ発生させます．そして，スキャンライン平面がポリゴンと交わった時に，その交線であるセグメントを投影面に，z 軸に平行に投影します．このようにして，次々にスキャンラインを下げていき，セグメントを順に表示すると，ポリゴンがディスプレイに表示されることになります．（図7）

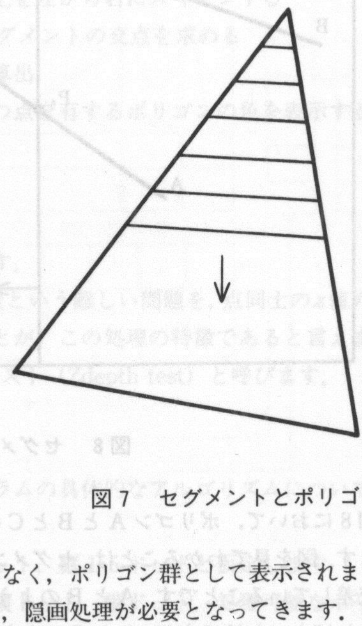


図7　セグメントとポリゴン

さて，実際にはポリゴンは一つだけではなく，ポリゴン群として表示されます．そして当然ながら，ポリゴンは重なり合い，隠画処理が必要となってきます．隠画処理は，手前側のポリコンを優先して表示させようという処理です．しかし，ポリゴンという面を扱い，優先度を判定すると，扱うデータが三次元になり，処理が複雑になってしまいます．そこで，実際の処理ではセグメント間の優先度の判定を行い，優先する，つまり正規透視座標系での z 値の小さい方のセグメントを表示していきます．すなわち，スキャンラインを上方から下方へと順に下げて，スキャンライン平面と交わるポリゴンがあれば，そのセグメントを計算し，何本かのセグメントを切り出します．そしてそれらのセグメントの間の z 値を比較し，値の小さな z 値を持つセグメントを表示します．この処理を各スキャンラインご

とに行ない，隠面処理を施します．

さて，セグメントの優先度の判定についてもう少し詳しい説明をします．図8はテレビの上方からスキャンライン平面，セグメントをながめた図です．

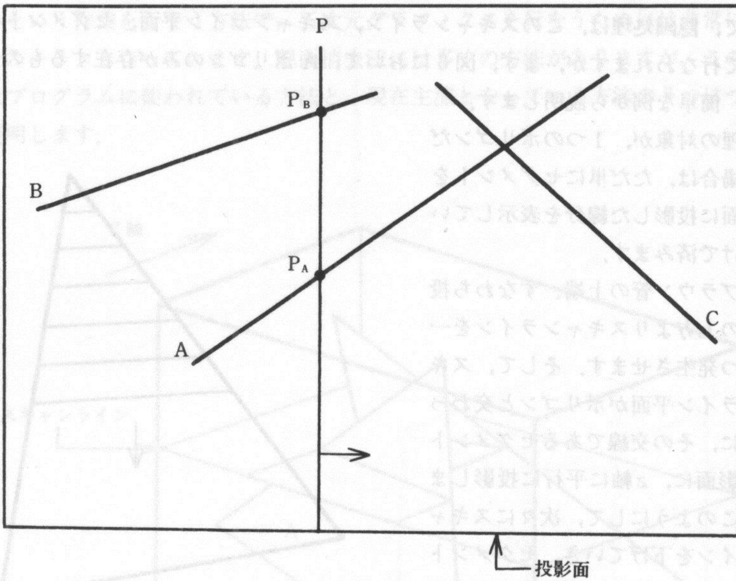


図8　セグメントの優先度判定

図8において，ポリゴンAとBとCのポリゴンがスキャンラインにかかっています．図を見てわかることは，セグメントAはBよりも優先度が高く，AとCは交差していることです．AとBのように交差しない場合は，どちらか一方を表示すればよいわけですが，AとCのように交差している場合はセグメントを2つに分離するなどの処理が必要となります．

ここで，使用する方法は，セグメントの優先度判定を行なわず，スキャンライン平面上に直線Pを引き，セグメントとの交点，P_A，P_B，P_Cの z 値を比較することにより，優先度を求めます．この直線Pは，スキャンラインが上から下へと走るのと同様に，左から右へと走り，そのつど，交点を求め，z値を比較し，zが最小値であるセグメントの色をスキャンライン上に点として表示して，1つのスキャンライン上のドットを埋めていきます．このように，スキャンライン上を左か

ら右へと走査する線分を考えることによって，三次元の隠れ面の問題が，一次元の点同士の z 値の比較という問題にまで単純化されるわけです．

なお，スキャンライン数は，出力するディスプレイの縦のドット数より多くとっても無駄となります．同様にスキャンライン上を左から右へと走査する際も，ディスプレイの横ドット数に合わせて，走査が行なわれます．

ではここで再び隠面処理についてまとめてみます．

プロセス１：スキャンラインを発生させる
　プロセス２：スキャンライン平面とポリゴンとの交線を求める
　プロセス３：スキャンライン上を左から右にスキャンする
　　プロセス４：線分Ｐとセグメントの交点を求める
　　　プロセス５：最小値算出
　　プロセス６：最小値をもつ点を有するポリゴンの色を表示する
　プロセス７：continue
プロセス８：continue

このようにして隠面処理は行なわれます．

本来は，三次元の面同士の優先度の比較という難しい問題を，点同士の z 値の比較という容易な問題に置きかえていることが，この処理の特徴であると言えましょう．この z 値の比較の作業を z デプステスト（Zdepth test）と呼びます．

○アルゴリズム

隠面消去を説明したところで，プログラムの具体的なアルゴリズムについて説明します．

まず．使用するディスプレイを横１３ドット，縦１５ドットと仮定します．

このアルゴリズムでは，ディスプレイの横ドット，縦ドット，処理可能なポリゴンの数の要素を持つ配列を用いて，ポリゴンが何番目のスキャンラインから何番目のスキャンラインまでかかっているのか，および，セグメントは左から何ドットから始まって何ドットで終わるのかという情報を書き込みます．そして次に，スキャンラインを発生させ，配列から何番のポリゴンがそのスキャンラインにかかっているかを読み出し，セグメントの位置，および長さを求め，横ドット数分の配列に書き込みます．最後にその配列の内容を１つずつ見ていき，１ドットごとの z デプステストを行ない、色を表示します．ではこれより具体的な例を掲げてプログラムのアルゴリズムを見てみましょう．

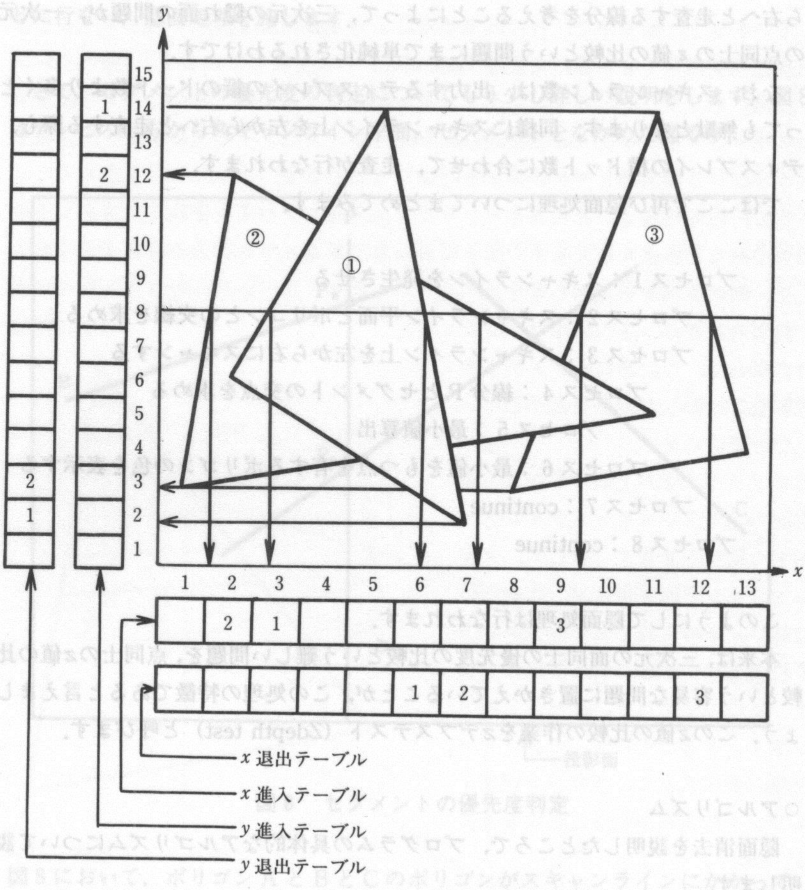


図９　各テーブルと画面との対応関係

上図はスクリーン上に映ったポリゴンと，このアルゴリズムで使用するx進入，x退出，y進入，y退出テーブルの内容を示したものです．ここで，ポリゴンのデータは正規透視座標系でのx値とy値を変換し，投影面左すみが（１，１）に，右すみが（１３，１５），（ディスプレイが１３×１５ドットで，アスペクト比が１の時）になるように変換されているとします．（なおz値は保存します）

さて，ポリゴンデータを変換した後，x進入，x退出，y進入，y退出の４つのテ

ーブルを用意します．y進入テーブルは，何番目のスキャンラインが，何番のポリゴンに最初に接するかという情報を持ち，ディスプレイの縦座標分の配列要素を持った，ポリゴンナンバが入る配列です．y退出は同様に何番目ののスキャンラインで、何番のポリゴンが抜け出るかという，同じくポリゴンナンバが入る配列です．図９の例ではy進入テーブルを見ると，１４番のスキャンラインに，No．１のポリゴンが進入し，１２番のスキャンラインでNo．２のポリゴンが退出することがわかります．

※スキャンラインはディスプレイのデータ転送の関係上、上から下へと走るので，最初のスキャンラインは１５番のスキャンラインで，順々に１つずつ下がります．パーソナルコンピュータのディスプレイは画面のどの場所からドットデータを送ろうともかまわないのですが，グラフィクス専用ディスプレイの場合，左上から１ドットずつデータを送る転送モードがあり，そのためにスキャンラインは上方から下方へと動きます．

y進入，y退出テーブルの具体的な計算としては，両者をそれぞれYentry，Yexitと名づけ，y_1～y_3をポリゴン頂点のｙ値とすると，

Yentry（IFIX（max（y_1，y_2，y_3）））＝Polygon Number
Yexit（IFIX（min（y_1，y_2，y_3）））＝Polygon Number

のように計算されます．

さて，このようにy進入，y退出テーブルが作られると，何番のスキャンラインに何番と何番のポリゴンが接しているかが容易に求められます．このスキャンラインに接しているポリゴンのリストを作成するためには，ポリゴンの総数分の要素を持つ配列，y占有テーブルを作ります．このテーブルは，現在のスキャンラインに何番のポリゴンが接しているかといった情報を記憶するもので，実際にはフラグが書き込まれます．この例ですとポリゴン総数は３ですから，Yoccupied(3)という３ビットの配列を取ります．（occupied：占有される）
そこで図９の例に従ってYoccupiedの働きを見ますと，

スキャンライン14番………Yoccupied(1)＝１，Yoccupied(3)＝１
スキャンライン12番………Yoccupied(2)＝１
スキャンライン３番………Yoccupied(2)＝０，Yoccupied(3)＝０
スキャンライン１番………Yoccupied(1)＝０

という動きをします．これによって，現在のスキャンラインにかかっているポリゴンのリストが，y占有テーブルのフラグのON，OFFによって表現されることになります．なお，ここで注意してほしいことは，y進入，y退出テーブルは，隠面消去処理の初めに作成され，処理の終了まで変化しないのに対し，y占有テーブルはスキャンラインが1つ下がるごとに更新されることです．

さて，今までの説明で，NO．3のポリゴンを意識的に省いてきたのですが，このポリゴンは，上辺がNO．1のポリゴンと同じy座標で，下辺がNO．2のポリゴンと同じy座標となっています．このため，y進入，y退出テーブルには，1および2というポリゴンナンバが代表して書き込まれていました．それではどのようにしてNO．3のポリゴンの情報を書き込むかという方法は，別にポリゴン数と同じ大きさのテーブルを作り，同一のスキャンラインに複数のポリゴンが進入あるいは退出することを示すデータを書き込むことによって行なわれます．このテーブルを以下に示し，その使用法を説明します．

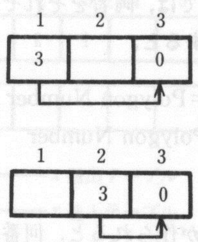


図10　Yentry′とYexit′

上図において，YentryはNO．1のポリゴンが進入するスキャンラインと同じスキャンラインにNO．3のポリゴンが進入することを表わしています．
同様にYexit′はNO．2のポリゴンが退出するスキャンラインと同じスキャンラインにNO．3のポリゴンが退出することを表わしています．この例は，図9と対応しているので，見較べて下さい．

実際のプログラムでは，ポリゴンの数は数万個にも達することがあり，一つのスキャンラインに何千個ものポリゴンが進入する状況が予想できますが，Yentry′，Yexit′はそのような状況にも対応できるようになっています．

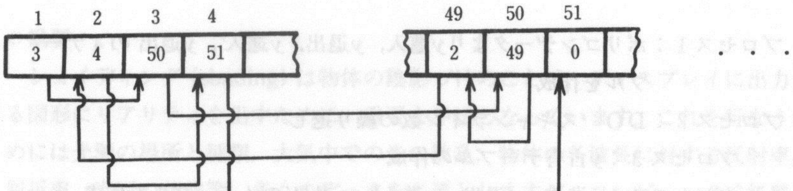


図11　リストの連結の様子

次ページの図のようにYentry′, Yexit′ともに，一つのスキャンラインに複数個のポリゴンがかかる時，イモズル式にポリゴンナンバを引き出すことができます．

2つのテーブルは，Yentry，Yexitの作成と同時に作成されます．そのため，実行の終了まで，その値は変わることがありません。

さて，画び図9に戻ってx進入テーブル，x退出テーブルの説明に移ります．図では8番のスキャンラインが発生しています．
x進入，x退出テーブルの意味はy進入，y退出テーブルの意味とほとんど変わりがありません．つまり，スキャンライン上を1ドットずつスキャンしていく際に，最初にポリゴンが接じ，そのドットに対応する要素にポリゴンナンバを書き込むテーブルがx進入テーブルです．そしてスキャンしていくドットから，最後に接し終わるドットに対応する要素にナンバを書き込むテーブルがx退出テーブルです．

y進入，y退出テーブルと同様に，1つのドットに複数のポリゴンの辺が状することも考えられるので，Xentry′，Xexit′という配列を同様にとり，さきほどの説明と同じ作業を行ないます．

さて，4つのテーブルが揃った後に，y占有テーブルと同じく，x占有テーブルを作成します．これは，スキャンライン上を1ドットずつスキャンするごとに更新されるテーブルで，そのドットを含むポリゴンナンバを参照するためのテーブルです．y占有テーブルと同様にフラグが書き込まれます．そしてx占有テーブルが作成された後には，そのリストに従って各ポリゴンのz値の算出を行ない，ポリゴン間のz値の比較：zデプステストを行ない，最小のz値を有するポリゴンの色をそのドットに表示します．

それで今まで述べてきたアルゴリズムをまとめてみます．

プロセス1：ポリゴンデータよりy進入，y退出，y進入′，y退出′の4テーブルを作成

プロセス2：DO　スキャンライン数の繰り返し

プロセス3：y占有テーブルの作成

プロセス4：x進入，x退出，x進入′，x退出′の4テーブルを作成

プロセス5：DO　横ドット数の繰り返し

プロセス6：x占有テーブルの作成

プロセス7：x占有テーブルのポリゴンリストにより，ポリゴン間のzデプステストを行なう．

プロセス8：CONTINUE

プロセス9：CONTINUE

○デプスバッファ

今まで説明を進めてきた，スキャンラインを使って隠面消去処理を行なう方法はスキャンラインアルゴリズムと呼ばれています．

しかし，現在はコンピュータの記憶容量が増えたため，zデプスバッファという隠画消去処理が主流になっています．これは画面上のドット数と同じ数の要素を持つ，実数型二次元配列を用意し，投影面に投影されたポリゴンのz値を，ポリゴンに含まれる画素に対応する配列に，入れます．このように，個々のポリゴンに対し，その占有する画素に対応する配列にz値を入れます．そしてもし，書き込もうとする配列に，他のポリゴンのz値が入っていたとすると，両者の小さい方の値を入れます．このようにして，総ポリゴン数の作業を終えると，この配列にはz値の最も小さい値が書き込まれ，簡単に隠画消去を行なうことができます．

この方法はzデプスバッファと呼ばれ，コンピュータの記憶容量が大きい場合は，ごく簡単に隠画消去を行なうことができるのですが，記憶容量の小さいパーソナルコンピュータ等などでは，比較的少ない記憶容量で実行可能なスキャンラインアルゴリズムに頼らざるを得ません．

隠画消去処理には他にも多くの方法があります．それらの詳しい内容については，参考文献を参照して下さい．

5．2．4 シェイディング

○概要

シェイディング (shading) は物体の陰影づけのことで，ディスプレイに出力する図形にリアリティを出すために，重要な手法となっています．これを行なうためには光源の場所と種類，大気中での光の散乱，物体の各波長に対する反射率，屈折率，表面の状態等，多くのデータを必要としますが，コンピュータで処理をするために，簡単な数式のモデルでシェイディングを表現します．

○数式モデル

シェイディングのモデルに必要な要素は，物体の一点に当たる光と，周囲からの散乱光と，入射光による散乱光と，正反射光 による輝度です．これら の条件により，表示する物体の一点の輝度を求め，ディスプレイに表示することが可能になります．

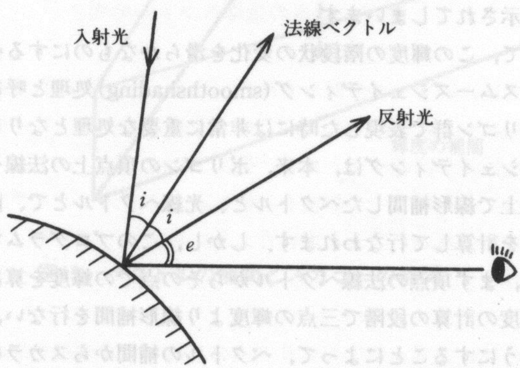


図12 シェイディングモデル

透明でない物体表面上の一点の輝度は，

$$L = a + b \cdot \cos i + c \cdot (\cos e)^{n}$$

という式でモデリングをしています．

この式のうちaは，物体の周囲の散乱光の度合いです．例えば同じ物体を宇宙空間に置いた時と，砂漠に置いたところでは，特に物体の光のあたらない部分の

輝度は異なるでしょう．したがってaは物体の輝度の最低値を保証するものと考えられます．

次のbcos iという式は，図11において，光による物体表面の乱反射光を計算します．

最後のc・(cos e)^{n}という式は，入射光が表面で反射された反射光のうち，目に入る輝度を表わすものです．

シェイディングはこのような式によってモデル化され，計算されます．なお，最近パーソナルコンピュータ上でも行なわれるようになったレイ・トレーシングは，物体表面から目に入って来る光を計算するのではなく，逆に目を出発点とした仮想の光路を仮定し，光路を物体表面で反射，屈折させながら光源にたどりつき，輝度を算出するものです．

○スムーズシェイディング

ポリゴンは平面であるため，そのままシェイディングを施すと，物体を分割したポリゴンの接ぎ面で輝度の境界ができてしまい，凸凹のある物体としてディスプレイに表示されてしまいます．

したがって，この輝度の階段状の変化を滑らかなものにする必要があります．この処理をスムーズシェイディング(smoothshading)処理と呼びます．この処理は物体をポリゴン群で表現した時には非常に重要な処理となります．

スムーズシェイディングは，本来，ポリゴンの頂点上の法線ベクトルを，ポリゴンの表面上で線形補間したベクトルと、光線ベクトルとで、前の数式モデルを使って輝度を計算して行なわれます．しかし，このプログラムでは，計算量を減らす目的で，まず頂点の法線ベクトルからその点での輝度を算出し，デプステストの後の輝度の計算の段階で三点の輝度より線形補間を行ない，表示を行ないます．このようにすることによって，ベクトルの補間からスカラの補間へと，処理は単純化されます．

図13にはこの法線ベクトルの補間法と，輝度の補間法を示します．

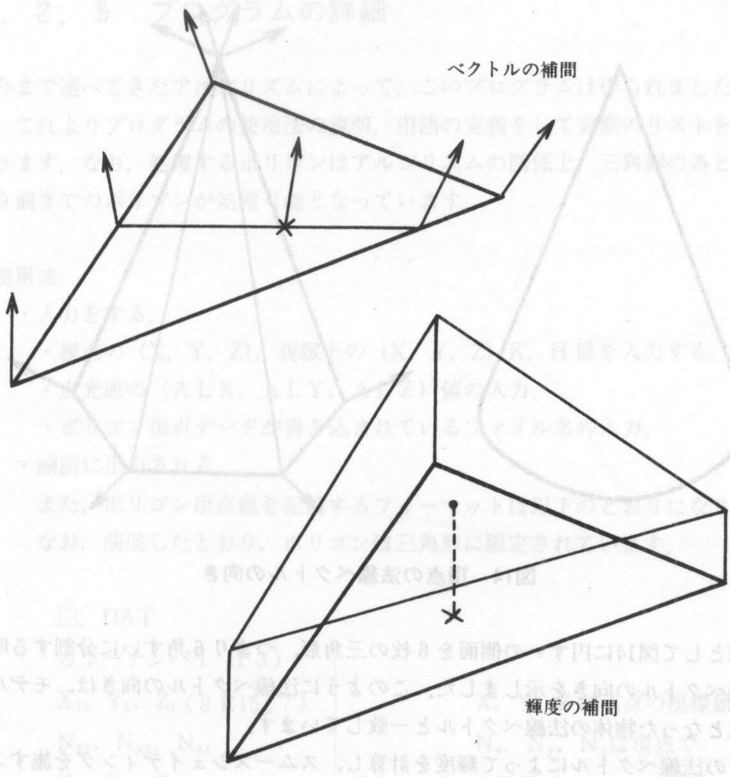


図13　ベクトルの補間とスカラーの補間

○法線ベクトル

ポリゴンは平面であるため，頂点の法線ベクトルは同じ方向を示しています．しかし，スムーズシェイディングを施すためには，ポリゴン本来の法線ベクトルとは異なった向きの法線ベクトルを頂点に設定する必要があります．

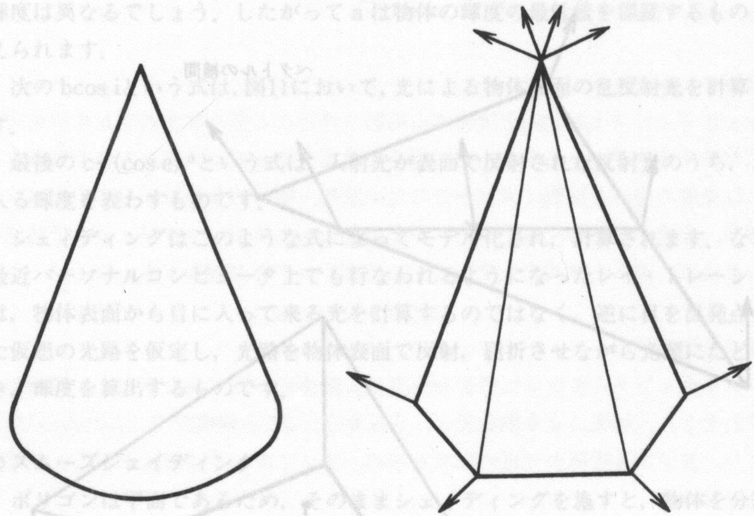


図14　頂点の法線ベクトルの向き

例として図14に円すいの側面を6枚の三角形，つまり6角すいに分割する時の法線ベクトルの向きを示しました．このように法線ベクトルの向きは，モデルの対象となった物体の法線ベクトルと一致しています．

この法線ベクトルによって輝度を計算し，スムーズシェイディングを施すことにより，物体の輪郭の凸凹など気にならない程度のリアリティのある三次元図形出力が得られます．

以上，このプログラムのアルゴリズムの基礎となる理論を説明してきました．ではこれより節を改め，プログラムの具体的な説明に入ることにします．

5．2．5　プログラムの詳細

今まで述べてきたアルゴリズムによって，このプログラムは作られました．では，これよりプログラムの使用法の説明，用語の定義をして実際のリストを見ていきます．なお，処理するポリゴンはアルゴリズムの関係上，三角形のみとし，49個までのポリゴンが処理可能となっています．

○使用法

・入力をする．

・視点の（X，Y，Z)，視線上の（X，Y，Z）K，H値を入力する．

・点光源の（ALX，ALY，ALZ）値の入力．

・ポリゴン項点データが書き込まれているファイル名の入力．

・画面に出力される．

また，ポリゴン項点値を記述するフォーマットは以下のとおりになります．なお，前述したとおり，ポリゴンは三角形に限定されています．

□．DAT

カラーナンバ1（I 3）	ポリゴン1	
X_1，Y_1，Z_1（3 E15．7）		X，Y，Zは項点の座標値
N_{x1}，N_{y1}，N_{z1}		N_x，N_y，N_zは項点で
X_2，Y_2，Z_2		の法線ベクトル．
N_{x2}，Y_{y2}，N_{z2}		
X_3，Y_3，Z_3		
N_{x3}，N_{y3}，N_{z3}		

ポリゴン2の記述

・

ポリゴン3の記述

：

○用語の定義，説明

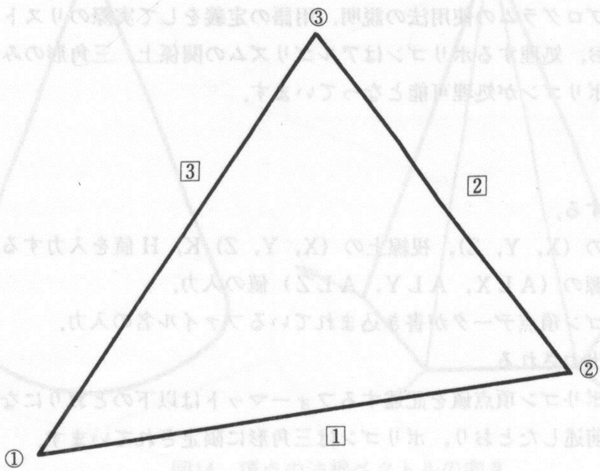


図15　頂点・陵線ナンバ

・ポリゴン陵線ナンバ1～3を図のように定めます．

・セグメントx値およびzデプスのインクリメンタル計算：セグメント，zデプスの計算には乗算を必要とし，特にzデプスの計算は画面のドット数ぶんの回数を行なうため，計算時間が増加します．

このためセグメントのx値では，スキャンラインが1つ下がるごとに増減するx値をDeltaXという変数に，zデプスでは，ドットが1つ右に動くごとに増減するz値をDeltaZとし，インクリメンタル計算を行なっています．

○プログラムの内容

以下より，プログラムの内容をリストに従って説明します．

ＦＯＲＴＲＡＮソースリストと見比べながら読んで下さい．

・COMMON文： サブルーチンCONSTANTに引き渡すための共通変数を宣言します．EFEA(Eyefrom Eyeat)は視点，視角，投影面の指定に，TVTPは$x-y-z$直交座標系から正規透視座標系へ変換する際のマトリクスデータ，MATは$x-y-z$直交座標系から視点座標系への変換マトリクスデータです．

・DIMENSION文： 処理に必要な配列の宣言．変数の意味については変数表で後述します．

・DATA文　： IROUNDという特殊な目的で使われる配列の内容です．

・定数設定

〈データ入力〉

・WRITE，READ文： 視点，視線上の一点，K，H，点光源の位置，ポリゴンデータが書き込まれているファイル名の入力を行ないます．

・入力されたポリゴンのファイルをOPENして5番に設定します．

・まず，ポリゴンの色ナンバを入力します．

・ポリゴン各項点についての3回のループ

・ポリゴン項点値の入力，なお，順序がZ，X，Yとなっている理由は，図形処理で用いる$x-y-z$直交座標と，一般的なZが上を向いた直交座標とでは，$x-y-z$のつけ方が異なるため，ここでデータをだまして読み込んでいます．またポリゴン項点値を入れる配列には，Point X～Point Zと，Poin XX～Poin ZZの2種類があります．このようにする理由は，次のクリッピング処理で，ポリゴン数が増減するためです．したがって最初にPoin XX～Poin ZZにデータを読み込んでおき，後にPoint X～Point Zに置き換えます．

なお，Shine DとShine，Z ColorとColorもこれを同じ関係にあります．

・次に項点上での法線ベクトルの入力．

・UNITという，ベクトルの単位ベクトル化を行なうサブルーチンをCALLする．したがって入力する法線ベクトルは単位ベクトルである必要はありません．

・この頂点に当たる光のベクトル成分算出．

・再びUNITを呼び，この光線ベクトルの単位化．

・ShineZという，輝度を求めるファンクション文により輝度を求め，ShineDという頂点の輝度を入れる配列に書き込む．

・Iは入力されたポリゴン数．

・ポリゴンが続く限り1番へ飛ぶ．

2　・入力機器の5番をCLOSEします．

・サブルーチンCONSTANTをCALLします．

このCONSTANTでは座標変換に必要なマトリクスデータが計算されます．なお，サブルーチン，ファンクション文については後に詳しく述べます．

〈パス1〉　視野外のポリゴンおよび，投影面をまたぐポリゴンはクリッピングを施す必要があります．このパス1は，座標変換と同時に，クリッピングを行ないます．

・Nはクリッピング後のポリゴン総数が入る変数．

・入力されたポリゴン総数分のループ開始．

・ZCOLORからCOLORにデータを移します．

・X，Y，Zに頂点値を移し，WRにCONSTANTで計算された変数を用いて，変換に必要な値をセット．

・WR＝0という事はx−y平面に水平に視線を向けた時に起こります．こうなりますと上式での計算は不可能となるので，ストップします．

・Point X～Point ZにX，Y，Zを変換して正規透視座標系のデータに変換して入力します．

・もし今計算した頂点が，視点と投影面の間にあるか，もしくは，視点の後ろにあれば，

・IFLUGという変数をインクリメント．Memo Jという，

J値つまり頂点ナンバをメモする変数をセット．

・ＩＦ文終わり．

1000 ・ＤＯ文終わり．

・A Max 1，A Min 1は，実数値データの最大，最小値を求める関数，ここでポリゴン各頂点のx，y方向の最大・最小値を求めます．

・もしポリゴンが視野内の，投影面より向こう側に全部入っているか，もしくは，ポリゴンが投影面をまたいでいるならば，

・もし，ポリゴンが投影面をまたいでいるならば，

・ＸＤＤ～ＺＤＤに$x-y-z$座標系から視点座標系に変換した座標値を入れる．（正規透視座標系ではないことに注意）

・もしＩＦＬＵＧ＝２，つまり２点が視点と投影面の間にあるならば，

・もし，視点座標系でのＺ値がＨより大きい頂点ならば，つまり，頂点ナンバをJJのＤＯ文で動かすうちに，投影面より向こう側の頂点があらわれたならば，Memo Jにその頂点ナンバを入れる．（以後，J＝１，３というようなループの場合は，ポリゴン頂点ナンバを動かしているものと解釈して下さい．）

・投影面をまたぐ線分のうち，一方の両端の座標を求める．

・媒介変数Ｔの計算．

・線分の不可視側の端を，投影面とその線分との交点に変更する．

・輝度をポリゴン陵線上で線形補間してＳＤＤに入れる

・もう一方の線分をクリッピングする

（処理は前と同じため，省略）

1025 ・ＤＯ　１０２５のループで，クリッピングされて，新しく計算された頂点座標を，Point X～Point Z，新しい輝度値をShineに入力する．

・ＥＬＳＥ文：　IFLUG＝１の場合，つまり１点が視点と投影面の間にある場合．

つづく

※クリッピングの具体的な処理を見てきたところで，このクリッピング処理をもう少し詳しく説明するために，ひとまずリストの説明を中断します．

プログラム上ではＩＦＬＵＧという変数がクリッピングの場合分けをしています．ＩＦＬＵＧは０～３の４つの値をとりますが，それぞれの場合を図16に示します．

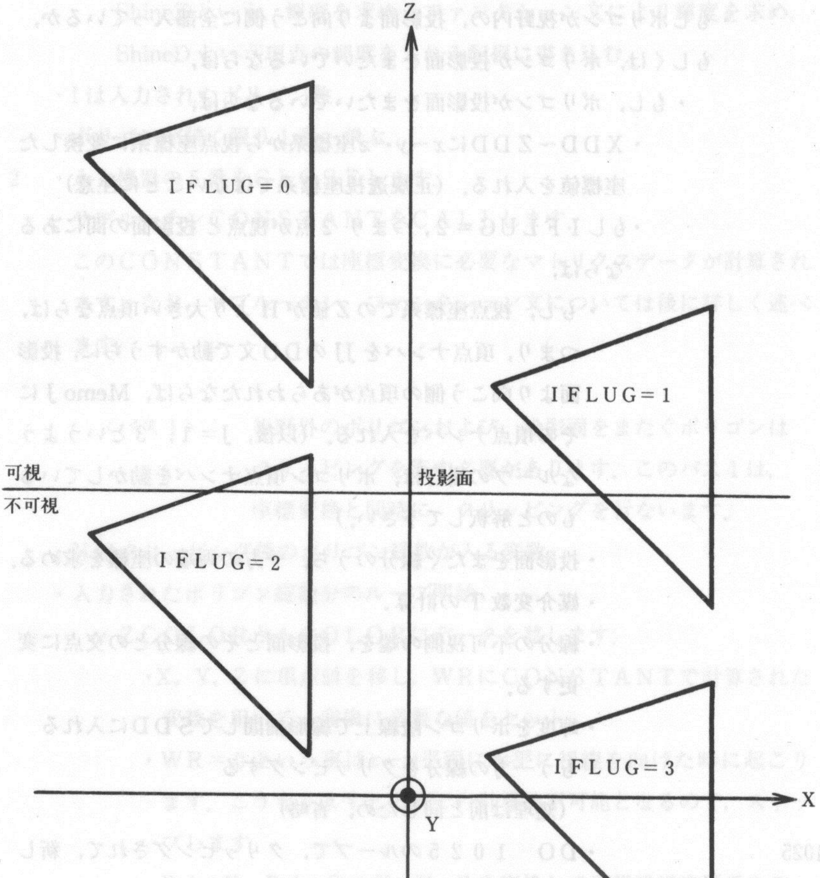


図16　ポリゴンの位置とＩＦＬＵＧ

図16は，視点座標系において，y軸の正の方向から真下に見たときの図です．投影面によって可視領域と不可視領域とに分けられるわけですが，ここで問題となることは，ポリゴンが投影面をまたぐ場合には(IFLUG＝1 or 2の場合)，ポリゴンを分割して可視領域のみのポリゴンに分けてやらねばなりません．

この処理を行いませんと，不可視領域内の頂点を正視透視座標系に変換する際に，Z値が1以上になるというおかしな値になってしまいます．

さて，以上により，クリッピング処理で問題となる場合はIFLUG＝1 or 2の場合であるという結論に達します．

ここで，IFLUG＝2 or 1の場合のポリゴンの分割方法を示します．

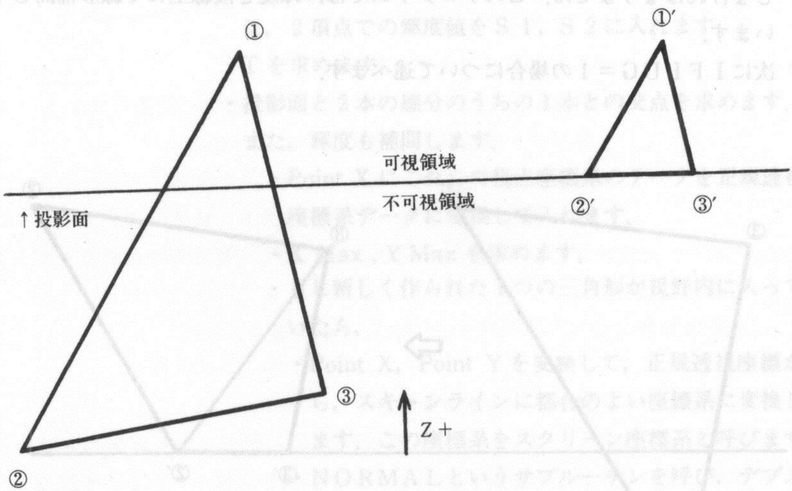


図17　IFLUG＝2の場合

IFLUG＝2の場合，可視領域に含まれるポリゴンは三角形のため，旧ポリゴンの図中②と③の頂点データを②′，③′に変更します．そのため，第一に，何番目の頂点が可視領域側に入っているのかをIF文によって判定します．図の例では1という頂点ナンバが得られるはずです．そこで，①－②の陵線，①－③の陵線と投影面との交点を求め，②′③′の2つの頂点データを算出します．そして旧ポリゴンデータを，この新ポリゴンのデータと置き換えます．

ここでプログラム中のIROUNDという配列について説明します．図15で，

1という，可視側の頂点ナンバーが求められると，それと対応する2，3という頂点ナンバが必要になります．このためには1という頂点ナンバに1を足して2を算出し，1を引いて3を算出すると便利です，

このため，ＩＲＯＵＮＤ（5）という配列をとり．

ＤＡＴＡ　ＩＲＯＵＮＤ／3，1，2，3，1　とデータを設定し，

頂点ナンバ1に1を加える操作では，ＩＲＯＵＮＤ(1＋1＋1),逆に1を引く操作では，ＩＲＯＵＮＤ(1－1＋1)という計算をします．なお，配列中の＋1とアンダーラインを施した部分は，配列の引き数が0にならないようにするための加算です．

さて，ポリゴン陵線のクリッピングと同時に，頂点も変わるため，輝度も変更しなければなりません．このプログラムでは，輝度を陵線上にて線形補間しています．

次にＩＦＬＵＧ＝1の場合について述べます．

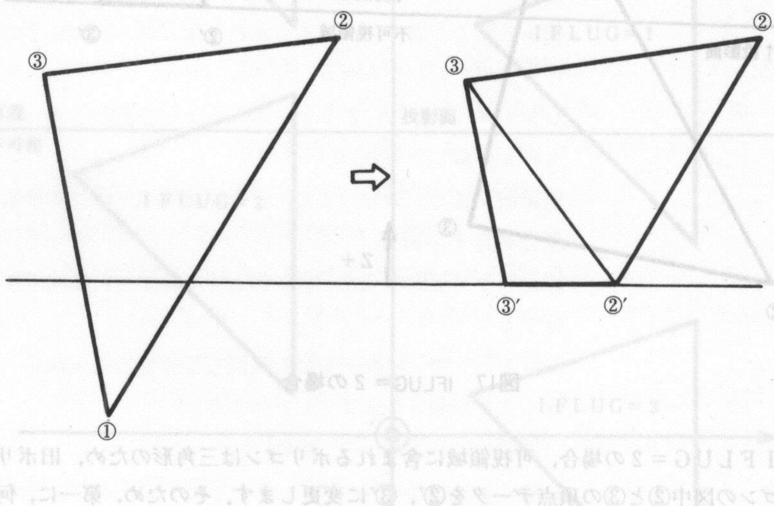


図18　IFLUG＝1の場合

ＩＦＬＵＧ＝1の場合，可視領域に含まれるポリゴンは四角形のために2つの三角形に分割する必要が生じます．しかし，基本的な考え方はＩＦＬＵＧ＝2

の場合と全く同様です.

このようにポリゴンの位置により，ポリゴン総数が増大するために，Poin XX ~Poin ZZというグループとPoint X~Point Zという2グループの配列を宣言し，読み込み時にはPoin XXに，クリッピング後はPoint Xに入れるという方法をとっています.

それでは再びプログラムの説明に戻ることにします.

・ELSE文：　IFLUG=1の場合
　・MEMOJには以前の処理によって視点と投影面との間にある頂点ナンバが入っているので，それを用いて2本の線分の片方ずつの頂点を求め，X1~Z1，X2~Z2に，2頂点での輝度値をS1，S2に入れます.
　・Tを求めます.
　・投影面と2本の線分のうちの1本との交点を求めます．また，輝度も補間します.
　　・Point Xにこれらの視点座標系のデータを正規透視座標系データに変換して入れます.
　　・X Max，Y Maxを求めます.
　　・もし新しく作られた1つの三角形が視野内に入っていたら，
　　　・Point X，Point Yを変換して，正規透視座標から，スキャンラインに都合のよい座標系に変換します．この座標系をスクリーン座標系と呼びます.
　　　・NORMALというサブルーチンを呼び，デプステストに必要となるポリゴンの平面式を求めます．また輝度を補間するために輝度平面の平面式も求めます.
　　　・DetaXという，ポリゴンのデプス計算の際に行なわれるインクリメンタル計算のための増分量を算出します.
　　　・Point X，Point Y=0の時，1にします．0にしておくと，見えないポリゴンとして処理されます.
　　　・X Max，X Min，Y Max，Y Minを求めます.
　　　　・DeltaXという，スキャンラインが1進むごと

に増分する．スキャンラインとセグメントとの交点のｘ値の増分量を求めます．

このDeltaxは，セグメント両端のｘ値のインクリメント計算に用います．

・Formul（３，N）＝０ならば微少量にします．

・N＝N＋１とし，つまり新しいポリゴンが１つ増加した情報を書き込みます．

・ＩＦＬＵＧ＝１の場合が続きます．

もう一つの三角形を作成します．

これ以下数行は今までと同じ処理です．

・ＣＯＬＯＲ（N）＝ＺＣＯＬＯＲ（I）で，ポリゴンの色をコピーします．

・ＩＦＬＵＧのＩＦ文終了．

・Point X～Point Zをスクリーン座標系に変換します．

・ポリゴン平面式および輝度平面式の算出．

・以下，さきほど述べた処理と同じものです．

・N＝N＋１で，Nをインクリメントします．

1005・ポリゴン総数分のループ終了．これにより全てのポリゴンはクリッピングされました．

・Nは実際のNより１大きいので，１を引きます．

〈ポリゴン陵線データ作成ブロック〉

ポリゴンとスキャンライン平面との交線であるセグメント両端のｘ値はスキャンラインが１つ更新されるごとに平面式から算出されるのではなく，ポリゴンが進入するスキャンラインでのセグメントデータに増分量を加えるインクリメンタル計算を行ないます．このブロックではそのための，最初のセグメントの両端のｘ値を求めます．

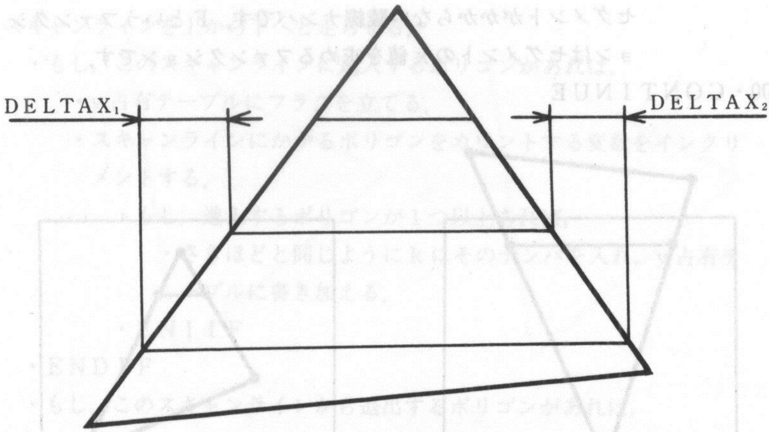


図19　セグメントのインクリメンタル計算

・ポリゴン数の繰り返し，

・もしポリゴンが画面上方にはみ出していたら，

・陵線と画面上方との交点のｘ値を求める．

・もし求めたｘ値が不適当ならば，

・Ryosenの内容を０にし，その陵線ナンバをひかえる．

・ＥＬＳＥ　適当ならば，

・後にDelta Ｘという増分量を足すことになるため，あらかじめ引いておく，

・ＥＬＳＥ：画面上方にはみ出していなかったら，

・ポリゴン上辺が水辺になっているか否かを調べるカウンタに０をセット．

・上辺が水辺になっているかどうかを調べる．

・もし，上辺が水平になっていたら．

・もし陵線No.１が水平だったら，陵線２と３を端とする最も上のセグメントの両端のｘ値を求め，Ryosenというｘ値を入れる配列にセットする．

・もし陵線No.２が水平だったら．

・もし陵線No.３が水平だったら．

・ＥＬＳＥ：普通の処理．ＩＲＥＭという配列の内容は，最初に

セグメントがかからない陵線ナンバです．Fというファンクションはセグメントのx値を求めるファンクションです．

1100・CONTINUE

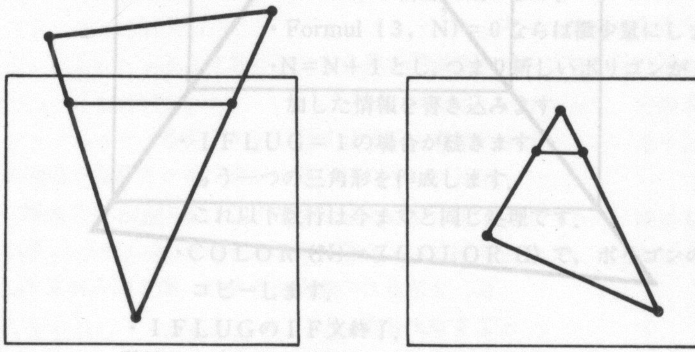


図20　画面からはみ出すポリゴン

〈y進入，y退出テーブル作成ブロック〉・ポリゴン数の繰り返し

・Y Max，Y Minを求め，y進入テーブルを参照できる値に正してI Y Max，I Y Minに入れる．

・もし，y進入テーブル（I Y En）に何も入っていなかったら

・そのままポリゴンナンバIを入れる．

・ELSE：ポリゴンナンバーがすでに入っていたら，

・すでに入っているポリゴンナンバを求めてkに入れる．

・y進入テーブル′（I Y Z E N）の内容より，ポリゴンナンバを順々に引き出し，最後のナンバを引き出す．

・I F文を抜けるとkには最後のポリゴンナンバが入っている．

・リストに現在のポリゴンナンバを追加する．

・E N D I F

・y退出テーブルについても同様の処理を行なう．

1200・CONTINUE

〈スキャンライン発生〉　スキャンラインを発生し，セグメント値を出し，デプステストを行なう最終ブロック

・スキャンラインを上から下へと走らせる．

・もし，このスキャンラインに進入するポリゴンがあれば，

・y占有テーブルにフラグを立てる．

・スキャンラインにかかるポリゴンをカウントする変数をインクリメントする．

・もし，進入するポリゴンが1つ以上あれば，

・さきほどと同じようにkにそのナンバを入れ，y占有テーブルに書き加える．

・ENIIF

・ENDIF

・もし，このスキャンラインから退出するポリゴンがあれば，

・以下，前の処理と同じ．

・ENDIF

・IYOCCOにスキャンラインを占めるポリゴン数をセットする．

・このIF文は，このプログラムを走らせていたコンピュータで使用できるWHILE文を直したもので，多少不自然となっています．

WHILE (I. NE. N)

・Iを1からポリゴン数まで回し，何番のポリゴンかを調べる．

・もし，Iが占有するポリゴンナンバであったならば，

・セグメント値のインクリメント．

・もし，セグメントの一端が新しい陵線にかかったならば，

・新陵線でのセグメントx値の算出を行なう．

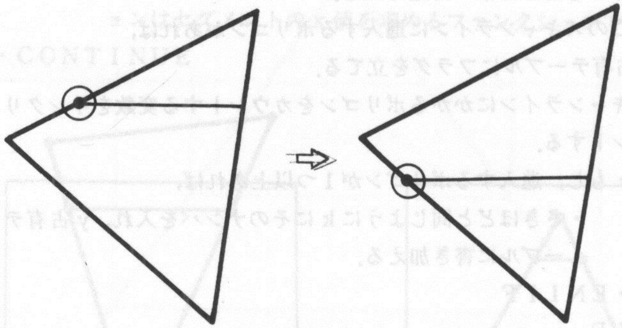


図21　新しいセグメントの生成

・セグメントの左端，右端値を求め，補正します．

・X進入テーブル，X退出テーブルを作ります．

この方法はYの場合と全く同じです．

1330　・WHILE文の代わりをするIF文の終了．

・x占有テーブルの代わりをするIPOINTに初期値をセットします．

y占有テーブルは上述のように使われていましたが，x占有テーブルは使われていません．この方法は，1ドットごとに，ポリゴン数分のサーチをするため，効率が悪いからです．そのため，I Pointという双方向ポインタを用いています．

例えばあるドットをNo. 4，6，2のポリゴンが占有する時．I Pointの内容は次のようになっています．

	0	1	2	3	4	5	6.... ポリゴンナンバ
1	0	5		0		3	
2	3	−1		5		1	

図22　I Point

まず（0，2）（横，縦）を見ると，3が書き込まれています．

そして（3，1）を見ますと，1と入り，双方は結合しています．

次に（3，2）を見ると，5が書き込まれ，以下（5，2），（1，1），（1，2）でENDマークである−1が入っています．

このようなＩPointerというリストを使用しています．

・スキャンライン上，x方向をスキャンする．

もし進入するポリゴンがあったなら，

・I Pointerにそのナンバを付け加える．

・ＩＰＥＮＤはリストの中の最後のポリゴンナンバ．

・このドットを占有するポリゴン数のカウンタをインクリメントする．

・進入するポリゴンの陵線上でのデプスを計算する．

このデプスもインクリメンタル計算を行ないます．すなわち，ポリゴンが進入する時にのみ，この平面式にてデプスを計算し，以降x方向に１進むごとにDelta Zという量をインクリメントし，加算にてデプスを計算します．

・進入するポリゴンが２個以上あれば，

・同様の処理を行ないます．

・ＥＮＤＩＦ

・ＩＸＥnに０を入れる．

・以後Ｘ退出についても同じ処理です．

・Ｚデプステストを行なうための値の設定．

1425 ・I Pointerにより，ドットを占有するポリゴンナンバーを引き出す．WHILE文の始まり．WHILE（J．NE．−1）

・デプスをインクリメントする．

・デプスの最小値を求める処理，Memo Jには最もデプスの小さいポリゴンナンバが入る．

1430 ・ここでＺデプステストが完了しています．輝度を計算し，色ナンバよりディスプレイに表示します．

・スキャンライン横方向のループ終了．

・配列の初期化

1300・スキャンラインのループ終了

以上でメインプログラムの説明は終わります.

○SUBROUTINE UNIT

このサブルーチンはベクトル (X, Y, Z) を単位ベクトル化するためのものです. 特に説明は要らないものと思われます.

○FUNCTION Shine

輝度を算出する. ここでは光線ベクトルと法線ベクトルとの内積をとるだけにとどまっています.

○SUBROUTINE CONSTANT

$x-y-z$直交座標系→正規透視座標系の変換マトリクスA11～A44および$x-y-z$直交座標系→視点座標の変換マトリクスAA11～AA44を求める.

○FUNCTION F

陵線とスキャンラインとの交点のx値を求めるサブルーチン.

○SUBROUTINE NORMAL

ポリゴン平面および輝度平面の平面式を求めるルーチンです.

以上でプログラムの説明を終わります.

5．2．6　変数名リスト

ではここにプログラムの主な変数名を掲げ，その意味を記します．

A11～A44	$x-y-z$⇒正規透視座標系への変換データ．
AA11～AA44	$x-y-z$⇒視点座標系への変換データ．
AK	視角を定めるパラメーターK．
ALX～ALZ	光源の座標．
COLOR（99）	ポリゴンの色ナンバ．
DELTAX（3，99）	セグメントのX増分値．
DELTAZ（99）	セグメントのZ増分値．
DEPTH（99）	ポリゴンのデプス．
FORMUL（4，99）	ポリゴン平面式の定数．
IFLUG	ポリゴンクリッピングに用いる変数．
IPOINT（2，100）	双方向ポインタ．
IROUND（5）	頂点値を参照するための配列．
IXEN（641）	X進入テーブル．
IXEX（641）	X退出テーブル．
IXOC（99）	X占有テーブル．
IXOCNU	ピクセスを占有するポリゴン数．
IXSCAN	スキャンライン上のドット番号．
IXZEN（99）	X進入テーブル′．
IXZEX（99）	X退出テーブル′．
IYEN（401）	Y進入テーブル．
IYEX（401）	Y退出テーブル．
IYOC（99）	Y占有テーブル．
IYOCNU	スキャンラインを占有するポリゴン数．
IYSCAN	スキャンライン番号．
IYZEN（99）	Y進入テーブル′．
IYZEX（99）	Y退出テーブル′．
N	ポリゴン総数．

POINTX	(3，99)	ポリゴンのX座標値.
POINTY	(3，99)	ポリゴンのY座標値.
POINTZ	(3，99)	ポリゴンのZ座標値.
POINXX	(3，99)	入力する際のポリゴンX座標値.
POINYY	(3，99)	入力する際のポリゴンX座標値
POINZZ	(3，99)	入力する際のポリゴンZ座標値
READFILFE		ポリゴンデータが書き込まれているファイル.
RYOSEN	(3，99)	セグメント両端のx値.
SFORMU	(4，99)	輝度平面の平面式係数
SHINE	(3，99)	輝度.
SHINED	(3，99)	輝度.
ZCOLOR	(99)	ポリゴンカラーナンバ.

5．2．7 プログラムの改造と応用

プログラムの改造として最も簡単な所は，輝度を計算するShineZというファンクション文でしょう．ここでは単に2つのベクトルの内積を計算しているだけですが，前に述べた数式モデルを使ったシェイディングを用いることによって本格的なシェイディングが可能となります．さらにここでは輝度を補間していますが，法線ベクトルを補間するようにしてもよいでしょう．

このプログラムではポリゴン数は49までとなっていますが，マイコンのメモリが許すならば配列中の（99）を（999）に，（100）を（1000）等に変更することにより，499個のポリゴンが処理可能となります．

応用として，アニメーションの作成が考えられます．筆者はこのプログラムを使い，一万個のポリゴンにて架空の都市データを作成し，都市内で視点を自由に動かし，8ミリカメラでコマ撮りを行なうことによってアニメーションを作成しました．また，三次元関数をポリゴンに分割して表示することもおもしろいと思われます．

○フォートランプログラム

```
C       3Dimensional graphics  By  Koji-Itou
        COMMON /EFEA/XF,YF,ZF,XA,YA,ZA,AK,H
        COMMON /TVTP/A11,A12,A13,A14,A21,A22,A23,A24,A31,A32,A33,A34
       1,A41,A42,A43,A44
        COMMON /MAT/AA11,AA12,AA13,AA14,AA21,AA22,AA23,AA24,AA31,AA32
       1,AA33,AA34,AA41,AA42,AA43,AA44
        DIMENSION PointX(3,99),PointY(3,99),PointZ(3,99)
        DIMENSION PoinXX(3,99),PoinYY(3,99),PoinZZ(3,99)
        DIMENSION Formul(4,99),SFormu(4,99),Shine(3,99),ShineD(3,99)
        DIMENSION IYEn(401),IYExit(401),IYZEN(99),IYZEX(99)
        DIMENSION IXEn(641),IXExit(641),IXZEN(99),IXZEX(99)
        DIMENSION XMax(99),XMin(99),YMax(99),YMin(99)
        DIMENSION Ryosen(3,99),DeltaX(3,99),DeltaZ(99)
        DIMENSION XDD(3),YDD(3),ZDD(3),SDD(3),Depth(99)
        INTEGER*2 IYOc(99),IXOc(99),IRound(5),IFlug
        INTEGER*2 ZColor(99),IREM(99)
        INTEGER*4 IP(2,2),IPoint(2,100),Color(99)
        CHARACTER*30 READFILE
        DATA IRound/3,1,2,3,1/,ICOUNT/0/,NEND/0/,I/1/,IRND/1/
C       " CLS " is an external function of machine language for
C       display clear and set 640*400 graphics mode.
        CALL CLS
        WRITE(*,'(A)') ' Shading? (Yes=1,No=0)'
        READ(*,'(I1)') IOK
C       *********  INPUT DATA  *******
        WRITE(*,'(A)') ' <----XFROM---><----YFROM---><----ZFROM--->'
        READ(*,'(3E15.7)') ZF,XF,YF
        WRITE(*,'(A)') ' <-----XAT-----><-----YAT-----><-----ZAT----->'
        READ(*,'(3E15.7)') ZA,XA,YA
        WRITE(*,'(A)') ' <------K------><------H------>'
        READ(*,'(2E15.7)') AK,H
        IF (IOK.EQ.1) THEN
          WRITE(*,'(A)') ' <----LightX---><----LightY---><----LightZ--->'
          READ(*,'(3E15.7)') ALZ,ALX,ALY
        ELSE
          ALX=123.456
          ALY=234.567
          ALZ=345.678
        ENDIF
        WRITE(*,'(A)') ' READ FILE NAME ?'
        READ(*,'(A)') READFILE
        OPEN(5,FILE=READFILE,STATUS='OLD')
1       READ(5,'(I3)',END=2) ZColor(I)
        DO 1000 J=1,3
          READ(5,'(3E15.7)') PoinZZ(J,I),PoinXX(J,I),PoinYY(J,I)
          READ(5,'(3E15.7)') ANZ,ANX,ANY
          CALL UNIT(ANX,ANY,ANZ)
          ALXX=ALX-PoinXX(J,I)
          ALYY=ALY-PoinYY(J,I)
          ALZZ=ALZ-PoinZZ(J,I)
          CALL UNIT(ALXX,ALYY,ALZZ)
          ShineD(J,I)=ShineZ(ALXX,ALYY,ALZZ,ANX,ANY,ANZ)
1000    CONTINUE
        I=I+1
        GOTO 1
2       CLOSE(5)
        CALL CONSTANT
        NN=I-1
        IYOCNu=0
        ICOUNT=0
        WRITE(*,'(A)') ' PASS 1 START'
C       ********  PASS 1  ********
C       deleate and create polygons
        N=1
        DO 1005 I=1,NN
          Color(N)=ZColor(I)
        IFlug=0
        DO 1010 J=1,3
          X=PoinXX(J,I)
          Y=PoinYY(J,I)
          Z=PoinZZ(J,I)
          WR=A13*X+A23*Y+A33*Z+A43+AK
           IF(WR.EQ.0) STOP
          PointX(J,N)=(A11*X+A21*Y+A31*Z+A41)/WR
          PointY(J,N)=(A12*X+A22*Y+A32*Z+A42)/WR
```

```
      PointZ(J,N)=(A13*X+A23*Y+A33*Z+A43)/WR
      IF((PointZ(J,N).LT.0).OR.(PointZ(J,N).GT.1)) THEN
C       if this point exist between Viewpoint and Vieplane
C       or behind the Viewpoint
        IFlug=IFlug+1
        MemoJ=J
      ENDIF
      Shine(J,I)=ShineD(J,I)
1010 CONTINUE
    XMax(N)=AMax1(PointX(1,N),PointX(2,N),PointX(3,N))
    XMin(N)=AMin1(PointX(1,N),PointX(2,N),PointX(3,N))
    YMax(N)=AMax1(PointY(1,N),PointY(2,N),PointY(3,N))
    YMin(N)=AMin1(PointY(1,N),PointY(2,N),PointY(3,N))
    IF(((XMax(N).GT.-1.5).AND.(YMin(N).GT.-1).AND.(XMin(N).LT.1.5)
   1  .AND.(YMin(N).LT.1).AND.(IFlug.EQ.0)).OR.(IFlug.NE.3)) THEN
     IF(IFlug.NE.0) THEN
      DO 1015 JJ=1,3
        XDD(JJ)=AA11*PoinXX(JJ,I)+AA21*PoinYY(JJ,I)+AA31*
   1            PoinZZ(JJ,I)+AA41
        YDD(JJ)=AA12*PoinXX(JJ,I)+AA22*PoinYY(JJ,I)+AA32*
   1            PoinZZ(JJ,I)+AA42
        ZDD(JJ)=AA13*PoinXX(JJ,I)+AA23*PoinYY(JJ,I)+AA33*
   1            PoinZZ(JJ,I)+AA43
        SDD(JJ)=ShineD(JJ,I)
1015  CONTINUE
      IF(IFlug.EQ.2) THEN
C       if two points exist between viepoint and vieplane
        DO 1020 JJ=1,3
          IF(ZDD(JJ).GT.H) MemoJ=JJ
1020    CONTINUE
        X1=XDD(IRound(MemoJ+2))
        X2=XDD(IRound(MemoJ+1))
        Y1=YDD(IRound(MemoJ+2))
        Y2=YDD(IRound(MemoJ+1))
        Z1=ZDD(IRound(MemoJ+2))
        Z2=ZDD(IRound(MemoJ+1))
        S1=SDD(IRound(MemoJ+2))
        S2=SDD(IRound(MemoJ+1))
        IF(Z1-Z2.EQ.0) THEN
          T=(Z1-H)/1.E-10
        ELSE
          T=(Z1-H)/(Z1-Z2)
        ENDIF
        XDD(IRound(MemoJ+2))=X1-T*(X1-X2)
        YDD(IRound(MemoJ+2))=Y1-T*(Y1-Y2)
        ZDD(IRound(MemoJ+2))=H
        SDD(IRound(MemoJ+2))=S1-T*(S1-S2)
C       an other line
        X1=XDD(IRound(MemoJ))
        Y1=YDD(IRound(MemoJ))
        Z1=ZDD(IRound(MemoJ))
        S1=SDD(IRound(MemoJ))
        IF(Z1-Z2.EQ.0) THEN
          T=(Z1-H)/1.E-10
        ELSE
          T=(Z1-H)/(Z1-Z2)
        ENDIF
        XDD(IRound(MemoJ))=X1-T*(X1-X2)
        YDD(IRound(MemoJ))=Y1-T*(Y1-Y2)
        ZDD(IRound(MemoJ))=H
        SDD(IRound(MemoJ))=S1-T*(S1-S2)
        DO 1025 JJ=1,3
          PointX(JJ,N)=H*XDD(JJ)/ZDD(JJ)/AK
          PointY(JJ,N)=H*YDD(JJ)/ZDD(JJ)/AK
          PointZ(JJ,N)=1-H/ZDD(JJ)
          Shine(JJ,N)=SDD(JJ)
1025    CONTINUE
      ELSE
C       iflug=1
C       one point exist between viepoint and vieplane
        X1=XDD(IRound(MemoJ+2))
        Y1=YDD(IRound(MemoJ+2))
        Z1=ZDD(IRound(MemoJ+2))
        S1=SDD(IRound(MemoJ+2))
        X2=XDD(IRound(MemoJ+1))
        Y2=YDD(IRound(MemoJ+1))
        Z2=ZDD(IRound(MemoJ+1))
        S2=SDD(IRound(MemoJ+1))
        IF(Z1-Z2.EQ.0) THEN
```

```
        T=(Z1-H)/1.E-10
      ELSE
        T=(Z1-H)/(Z1-Z2)
      ENDIF
      X11=X1-T*(X1-X2)
      Y11=Y1-T*(Y1-Y2)
C     Z11=H
      S11=S1-T*(S1-S2)
      XDD(IRound(MemoJ+1))=X11
      YDD(IRound(MemoJ+1))=Y11
      ZDD(IRound(MemoJ+1))=H
      SDD(IRound(MemoJ+1))=S11
      DO 1030 JJ=1,3
        PointX(JJ,N)=H*XDD(JJ)/ZDD(JJ)/AK
        PointY(JJ,N)=H*YDD(JJ)/ZDD(JJ)/AK
        PointZ(JJ,N)=1-H/ZDD(JJ)
        Shine(JJ,N)=SDD(JJ)
1030  CONTINUE
      XMax(N)=AMax1(PointX(1,N),PointX(2,N),PointX(3,N))
      XMin(N)=AMin1(PointX(1,N),PointX(2,N),PointX(3,N))
      YMax(N)=AMax1(PointY(1,N),PointY(2,N),PointY(3,N))
      YMin(N)=AMin1(PointY(1,N),PointY(2,N),PointY(3,N))
      IF((XMax(N).GT.-1.5).AND.(YMax(N).GT.-1).AND.(XMin(N).LT.
     1 1.5).AND.(YMin(N).LT.1)) THEN
      DO 1035 J=1,3
        PointX(JJ,N)=PointX(JJ,N)*212+320
        PointY(JJ,N)=PointY(JJ,N)*199+200
1035    CONTINUE
        CALL NORMAL(PointX(1,N),PointY(1,N),PointZ(1,N),
     1  PointX(2,N),PointY(2,N),PointZ(2,N),PointX(3,N),
     1  PointY(3,N),PointZ(3,N),FormuL(1,N),FormuL(2,N),
     1  FormuL(3,N),FormuL(4,N))
        CALL NORMAL(PointX(1,N),PointY(1,N),Shine(1,N),
     1  PointX(2,N),PointY(2,N),Shine(2,N),PointX(3,N),
     1  PointY(3,N),Shine(3,N),SFormu(1,N),SFormu(2,N),
     1  SFormu(3,N),SFormu(4,N))
        IF(Formul(3,N).EQ.0) THEN
          DeltaZ(N)=-Formul(1,N)/(Formul(3,N)+1.E-10)
        ELSE
          DeltaZ(N)=-Formul(1,N)/(Formul(3,N))
        ENDIF
        DO 1040 JJ=1,3
          IF(PointX(JJ,N).EQ.0) PointX(JJ,N)=1
          IF(PointY(JJ,N).EQ.0) PointY(JJ,N)=1
1040    CONTINUE
        XMax(N)=AMax1(PointX(1,N),PointX(2,N),PointX(3,N))
        XMin(N)=AMin1(PointX(1,N),PointX(2,N),PointX(3,N))
        YMax(N)=AMax1(PointY(1,N),PointY(2,N),PointY(3,N))
        YMin(N)=AMin1(PointY(1,N),PointY(2,N),PointY(3,N))
        DO 1045 JJ=1,3
          IF(PointY(IRound(JJ+2),N)-PointY(JJ,N).EQ.0) THEN
            DeltaX(JJ,N)=-(PointX(IRound(JJ+2),N)-PointX(JJ,N))/
     1      (PointY(IRound(JJ+2),N)-PointY(JJ,N)+1.E-10)
          ELSE
            DeltaX(JJ,N)=-(PointX(IRound(JJ+2),N)-PointX(JJ,N))/
     1      (PointY(IRound(JJ+2),N)-PointY(JJ,N))
          ENDIF
1045    CONTINUE
        IF(Formul(3,N).EQ.0) Formul(3,N)=Formul(3,N)+1.E-10
        N=N+1
      ENDIF
      X1=XDD(IRound(MemoJ))
      Y1=YDD(IRound(MemoJ))
      Z1=ZDD(IRound(MemoJ))
      S1=SDD(IRound(Memoj))
      IF(Z1-Z2.EQ.0) THEN
        T=(Z1-H)/1.E-10
      ELSE
        T=(Z1-H)/(Z1-Z2)
      ENDIF
      X22=X1-T*(X1-X2)
      Y22=Y1-T*(Y1-Y2)
      Z22=H
      S22=S1-T*(S1-S2)
      XDD(IRound(MemoJ+1))=X22
      YDD(IRound(MemoJ+1))=Y22
      ZDD(IRound(MemoJ+1))=H
      SDD(IRound(MemoJ+1))=S22
      XDD(IRound(MemoJ+2))=X11
```

```
          MemoJ=J
        ENDIF
1110    CONTINUE
        IF(Memo.EQ.2) THEN
          IF(PointY(1,I).EQ.PointY(2,I)) THEN
            Ryosen(2,I)=F(PointX(2,I),PointY(2,I),PointX(3,I),
     1      PointY(3,I),PointY(2,I))-DeltaX(2,I)
            Ryosen(3,I)=F(PointX(3,I),PointY(3,I),PointX(1,I),
     1      PointY(1,I),PointY(1,I))-DeltaX(3,I)
            IREM(I)=1
          ENDIF
          IF(PointY(2,I).EQ.PointY(3,I)) THEN
            Ryosen(1,I)=F(PointX(1,I),PointY(1,I),PointX(2,I),
     1      PointY(2,I),PointY(2,I))-DeltaX(1,I)
            Ryosen(3,I)=F(PointX(3,I),PointY(3,I),PointX(1,I),
     1      PointY(1,I),PointY(3,I))-DeltaX(3,I)
            IREM(I)=2
          ENDIF
          IF(PointY(3,I).EQ.PointY(1,I)) THEN
            Ryosen(1,I)=F(PointX(1,I),PointY(1,I),PointX(2,I),
     1      PointY(2,I),PointY(1,I))-DeltaX(1,I)
            Ryosen(2,I)=F(PointX(2,I),PointY(2,I),PointX(3,I),
     1      PointY(3,I),PointY(3,I))-DeltaX(2,I)
            IREM(I)=3
          ENDIF
        ELSE
          Ryosen(IRound(MemoJ),I)=F(PointX(IRound(MemoJ),I),
     1    PointY(IRound(MemoJ),I),PointX(MemoJ,I),PointY(MemoJ,I),
     1    PointY(MemoJ,I))-DeltaX(IRound(MemoJ),I)
          Ryosen(MemoJ,I)=F(PointX(IRound(MemoJ+2),I),
     1    PointY(IRound(MemoJ+2),I),PointX(MemoJ,I),PointY(MemoJ,I),
     1    PointY(MemoJ,I))-DeltaX(MemoJ,I)
          IREM(I)=IRound(MemoJ+2)
        ENDIF
      ENDIF
1100  CONTINUE
      WRITE(*,'(A)') ' PASS 2 END'

C     ****** caluclating of iyentry,iyexit *****
      DO 1200 I=1,N
        IF(YMax(I).GT.400) THEN
          IYMax=400
        ELSE
          IF(YMax(I).LT.1) THEN
            IYMax=1
          ELSE
          IF(YMax(I).LT.1) THEN
            IYMax=1
          ELSE
            IYMax=IFIX(YMax(I))
          ENDIF
        ENDIF
        IF(YMin(I).LT.1) THEN
          IYMin=1
        ELSE
          IF(YMin(I).GT.400) THEN
            IYMin=400
          ELSE
            IYMin=IFIX(YMin(I))
          ENDIF
        ENDIF
        IF(IYEn(IYMax).EQ.0) THEN
          IYEn(IYMax)=I
        ELSE
          K=IYEn(IYMax)
1205      IF(IYZEN(K).EQ.0) GOTO1210
            K=IYZEN(K)
          GOTO1205
1210      IYZEN(K)=I
        ENDIF
        IF(IYExit(IYMin).EQ.0) THEN
          IYExit(IYMin)=I
        ELSE
          K=IYExit(IYMin)
1215      IF(IYZEX(K).EQ.0) GOTO 1220
            K=IYZEX(K)
          GOTO1215
1220      IYZEX(K)=I
```

```
        YDD(IRound(MemoJ+2))=Y11
        ZDD(IRound(MemoJ+2))=Z11
        SDD(IRound(MemoJ+2))=S11
        DO 1050 JJ=1,3
          PointX(JJ,N)=H*XDD(JJ)/ZDD(JJ)/AK
          PointY(JJ,N)=H*YDD(JJ)/ZDD(JJ)/AK
          PointZ(JJ,N)=1-H/ZDD(JJ)
          Shine(JJ,N)=SDD(JJ)
1050    CONTINUE
        Color(N)=ZColor(I)
      ENDIF
    ENDIF
    DO 1055 JJ=1,3
      PointX(JJ,N)=PointX(JJ,N)*212+320
      PointY(JJ,N)=PointY(JJ,N)*199+200
1055    CONTINUE
    CALL NORMAL(PointX(1,N),PointY(1,N),PointZ(1,N),
   1 PointX(2,N),PointY(2,N),PointZ(2,N),PointX(3,N),
   1 PointY(3,N),PointZ(3,N),FormuL(1,N),FormuL(2,N),
   1 FormuL(3,N),FormuL(4,N))
    CALL NORMAL(PointX(1,N),PointY(1,N),Shine(1,N),
   1 PointX(2,N),PointY(2,N),Shine(2,N),PointX(3,N),
   1 PointY(3,N),Shine(3,N),SFormu(1,N),SFormu(2,N),
   1 SFormu(3,N),SFormu(4,N))
    IF(Formul(3,N).EQ.0) THEN
      DeltaZ(N)=-(Formul(1,N))/(Formul(3,N)+1.E-10)
    ELSE
      DeltaZ(N)=-Formul(1,N)/Formul(3,N)
    ENDIF
    DO 1060 JJ=1,3
      IF(PointX(JJ,N).EQ.0) PointX(JJ,N)=1
      IF(PointY(JJ,N).EQ.0) PointY(JJ,N)=1
1060    CONTINUE
    XMax(N)=AMax1(PointX(1,N),PointX(2,N),PointX(3,N))
    XMin(N)=AMin1(PointX(1,N),PointX(2,N),PointX(3,N))
    YMax(N)=AMax1(PointY(1,N),PointY(2,N),PointY(3,N))
    YMin(N)=AMin1(PointY(1,N),PointY(2,N),PointY(3,N))
    DO 1065 JJ=1,3
      IF(PointY(IRound(JJ+2),N)-PointY(JJ,N).EQ.0) THEN
        DeltaX(JJ,N)=-(PointX(IRound(JJ+2),N)-PointX(JJ,N))/
   1    (PointY(IRound(JJ+2),N)-PointY(JJ,N)+1.E-10)
      ELSE
        DeltaX(JJ,N)=-(PointX(IRound(JJ+2),N)-PointX(JJ,N))/
   1    (PointY(IRound(JJ+2),N)-PointY(JJ,N))
      ENDIF
1065    CONTINUE
    IF(Formul(3,N).EQ.0) Formul(3,N)=Formul(3,N)+1.E-10
    N=N+1
  ENDIF
1005  CONTINUE
  N=N-1
  WRITE(*,'(A)') ' PASS 1 END'
  WRITE(*,'(A)') ' PASS 2 START'
C ********  Ryosen  *******
  DO 1100 I=1,N
    IF(YMax(I).GT.400) THEN
      DO 1105 J=1,3
        IF(PointX(J,I).NE.PointX(IRound(J+2),I)) THEN
          Ryosen(J,I)=(400-PointY(J,I))/
   1      (PointY(J,I)-PointY(IRound(J+2),I)+1.E-10)*
   1      (PointX(J,I)-PointX(IRound(J+2),I))+PointX(J,I)
        ELSE
          Ryosen(J,I)=PointX(J,I)
        ENDIF
        SIGN1=400-PointY(J,I)
        SIGN2=400-PointY(IRound(J+2),I)
        IF(SIGN1.EQ.SIGN(SIGN1,SIGN2)) THEN
C       if this line does not have an edge of segment.
          Ryosen(J,I)=0
          IREM(I)=J
        ELSE
          Ryosen(J,I)=Ryosen(J,I)-DeltaX(J,I)
        ENDIF
1105      CONTINUE
    ELSE
      Memo=0
      DO 1110 J=1,3
        IF(YMax(I).EQ.PointY(J,I)) THEN
          Memo=Memo+1
```

```
         ENDIF
 1200  CONTINUE
       WRITE(*,'(A)') ' PASS 3 START'
C      ******  large loop of iyscan  *****
      DO 1300 IYSCAN=400,1,-1
C       ******  caluclating of iyoc  ******
        IF(IYEn(IYSCAN).NE.0) THEN
          IYOc(IYEn(IYSCAN))=1
          IYOCNu=IYOCNu+1
          IF(IYZEN(IYEn(IYSCAN)).NE.0) THEN
            K=(IYEn(IYSCAN))
            IYOCNu=IYOCNu-1
 1305       IF(K.EQ.0) GOTO 1310
              IYOc(K)=1
              IYOCNu=IYOCNu+1
              K=IYZEN(K)
            GOTO1305
 1310     ENDIF
        ENDIF
        IF(IYExit(IYSCAN).NE.0) THEN
          IYOc(IYExit(IYSCAN))=0
          IYOCNu=IYOCNu-1
          IF(IYZEX(IYExit(IYSCAN)).NE.0) THEN
            K=IYExit(IYSCAN)
            IYOCNu=IYOCNu+1
 1315       IF(K.EQ.0) GOTO 1320
              IYOc(K)=0
              IYOCNu=IYOCNu-1
              K=IYZEX(K)
            GOTO1315
 1320     ENDIF
        ENDIF
        WRITE(*,'(A,2I3)') ' SCANLINE .........  ',IYSCAN,IYOCNU
        I=0
        IYOCCo=IYOCNu
        IF(IYOCNu.EQ.0) I=N
 1325   IF(I.EQ.N) GOTO 1330
          I=I+1
          IF(IYOc(I).EQ.1) THEN
            DO 1340 J=1,3
              IF(Ryosen(J,I).NE.0) Ryosen(J,I)=Ryosen(J,I)+DeltaX(J,I)
 1340       CONTINUE
            DO 1350 J=1,3
              IF((IYSCAN.LT.PointY(J,I)).AND.(IYSCAN.LT.
     1        PointY(IRound(J+2),I)).AND.(Ryosen(J,I).NE.0)) THEN
                K=IREM(I)
                IREM(I)=J
                Ryosen(K,I)=(IYSCAN-PointY(K,I))/
     1          (PointY(K,I)-PointY(IRound(K+2),I)+1.E-10)*
     1          (PointX(K,I)-PointX(IRound(K+2),I))+PointX(K,I)
                Ryosen(J,I)=0
              ENDIF
 1350       CONTINUE
C           ******  finding of xmax,xmin  ******
            ZXMin=30000
            ZXMax=-30000
            DO 1355 J=1,3
              IF((Ryosen(J,I).LT.ZXMin).AND.(Ryosen(J,I).NE.0)) ZXMin
     1        =Ryosen(J,I)
              IF(Ryosen(J,I).GT.ZXMax) ZXMax=Ryosen(J,I)
 1355       CONTINUE
            IF(ZXMax.GT.640) THEN
              IXMax=640
            ELSE
              IF(ZXMax.LT.0) THEN
                IXMax=1
              ELSE
                IXMax=IFIX(ZXMax)+1
              ENDIF
            ENDIF
            IF(ZXMin.LT.1) THEN
              IXMin=1
            ELSE
```

```
          IF(ZXMin.GT.640) THEN
            IXMin=640
          ELSE
            IXMin=IFIX(ZXMin)
          ENDIF
        ENDIF
        IF(IXEn(IXMin).EQ.0) THEN
          IXEn(IXMin)=I
        ELSE
          K=IXEn(IXMin)
1360      IF(IXZEN(K).EQ.0) GOTO 1365
            K=IXZEN(K)
          GOTO1360
1365      IXZEN(K)=I
        ENDIF
        IF(IXExit(IXMax).EQ.0) THEN
          IXExit(IXMax)=I
        ELSE
          K=IXExit(IXMax)
1370      IF(IXZEX(K).EQ.0) GOTO 1375
            K=IXZEX(K)
          GOTO1370
1375      IXZEX(K)=I
        ENDIF
      ENDIF
      GOTO 1325
C     ******  WHILE STATMENT END  ******
1330  IXocNU=0
      IPoint(1,0)=0
      IPoint(2,0)=-1
      IPEND=0
C     ******  large loop of ixscan  ******
      DO 1400 IXSCAN=1,640
C       ******  caluclating of ipointer  ******
        IF(IXEn(IXSCAN).NE.0) THEN
          Number=IXEn(IXSCAN)
          IPoint(2,IPEND)=Number
          IPoint(1,Number)=IPEND
          IPoint(2,Number)=-1
          IPEND=Number
          IXocNU=IXocNU+1
          IF(Formul(3,Number).NE.0) THEN
            Depth(Number)=-(IXSCAN*Formul(1,Number)+IYSCAN*Formul
     1      (2,Number)+Formul(4,Number))/Formul(3,Number)-DELTAZ
     1      (Number)
          ELSE
            Depth(Number)=1
          ENDIF
          IF(IXZEN(Number).NE.0) THEN
            K=IXZEN(Number)
1405        IF(K.EQ.0) GOTO 1410
              IPoint(2,IPEND)=K
              IPoint(1,K)=IPEND
              IPoint(2,K)=-1
              IPEND=K
              IXocNU=IXocNU+1
              IF(Formul(3,Number).NE.0) THEN
                Depth(K)=-(IXSCAN*Formul(1,K)+IYSCAN*Formul(2,K)
     1          +Formul(4,K))/Formul(3,K)-DELTAZ(K)
              ELSE
                Depth(K)=1
              ENDIF
              K=IXZEN(K)
            GOTO1405
1410      ENDIF
        ENDIF
        IXEn(IXSCAN)=0
        IF(IXExit(IXSCAN).NE.0) THEN
          Number=IXExit(IXSCAN)
          IF(IPoint(2,Number).NE.-1) THEN
            IPoint(2,IPoint(1,Number))=IPoint(2,Number)
            IPoint(1,IPoint(2,Number))=IPoint(1,Number)
          ELSE
            IPEND=IPoint(1,Number)
            IPoint(2,IPEND)=-1
          ENDIF
          IPoint(1,Number)=0
          IPoint(2,Number)=0
          IXocNU=IXocNU-1
```

```
      Depth(Number)=-10000
      IF(IXZEX(Number).NE.0) THEN
        K=IXZEX(Number)
1415    IF(K.EQ.0) GOTO 1420
          IF(IPoint(2,K).NE.-1) THEN
            IPoint(2,IPoint(1,K))=IPoint(2,K)
            IPoint(1,IPoint(2,K))=IPoint(1,K)
          ELSE
            IPEND=IPoint(1,K)
            IPoint(2,IPEND)=-1
          ENDIF
          IPoint(1,K)=0
          IPoint(2,K)=0
          IXocNU=IXocNU-1
          Depth(K)=-10000
          K=IXZEX(K)
        GOTO 1415
1420    ENDIF
      ENDIF
      IXExit(IXSCAN)=0
      Amin=10000
      MemoJ=0

      S=0
      J=IPoint(2,0)
1425  IF(J.EQ.-1) GOTO 1430
        Depth(J)=Depth(J)+DELTAZ(J)
C       ****** depth test  *****
        IF((Depth(J).LT.Amin).AND.(Depth(J).GE.0).AND.(Depth(J)
     1  .LE.1)) THEN
C         Memoj is a polygon number of minimum depth
          MemoJ=J
          Amin=Depth(J)
        ENDIF
        J=IPoint(2,J)
        GOTO 1425
1430  IF(MemoJ.NE.0) THEN
        IF(IOK.EQ.1) THEN
          IRND=MOD(125*IRND,8192)
          RND=FLOAT(IRND)/8192.
          S=-(SFormu(1,MemoJ)*FLOAT(IXSCAN)+SFormu(2,MemoJ)*FLOAT
     1    (IYSCAN)+SFormu(4,MemoJ))/SFormu(3,MemoJ)*100
C         " PSET " is also external function of machine language
C         for set pixel color.
          CALL PSET(IXSCAN-1,400-IYSCAN,IBRIGH(RND,S,Color(MemoJ)))
        ELSE
          CALL PSET(IXSCAN-1,400-IYSCAN,Color(MemoJ))
        ENDIF
      ENDIF
1400  CONTINUE
C     ****** ixscan loop continue  ******
      DO 1500 II=1,N
        IXZEX(II)=0
        IXOc(II)=0
1500  CONTINUE
C     ****** iyscan loop continue  ******
1300  CONTINUE
      END

      SUBROUTINE UNIT(X,Y,Z)
      A=SQRT(X*X+Y*Y+Z*Z)
      X=X/A
      Y=Y/A
      Z=Z/A
      RETURN
      END

      FUNCTION ShineZ(X1,Y1,Z1,X2,Y2,Z2)
      ShineZ=X1*X2+Y1*Y2+Z1*Z2
      RETURN
      END

      SUBROUTINE CONSTANT
      COMMON /EFEA/Xfrom,Yfrom,Zfrom,Xat,Yat,Zat,AK,H
      COMMON /TVTP/A11,A12,A13,A14,A21,A22,A23,A24,A31,A32,A33,A34
     1,A41,A42,A43,A44
      COMMON /Mat/AA11,AA12,AA13,AA14,AA21,AA22,AA23,AA24,AA31,AA32
```

```
     1,AA33,AA34,AA41,AA42,AA43,AA44
C     *******   MatRIX DatA  *******
      XSUB=Xfrom-Xat
      YSUB=Yfrom-Yat
      ZSUB=Zfrom-Zat
      A=SQRT(XSUB*XSUB+ZSUB*ZSUB)
      B=SQRT(XSUB*XSUB+YSUB*YSUB+ZSUB*ZSUB)
      IF(A.NE.0) THEN
        COSA=ZSUB/A
        SINA=-XSUB/A
      ELSE
        COSA=0
        SINA=0
      ENDIF
      IF(B.NE.0) THEN
        COSB=A/B
        SINB=YSUB/B
      ELSE
        COSB=0
        SINB=0
      ENDIF
C     ******
      AA11=COSA
      AA12=SINA*SINB
      AA13=SINA*COSB
      AA14=0
      AA21=0
      AA22=COSB
      AA23=-SINB
      AA24=0
      AA31=SINA
      AA32=-COSA*SINB
      AA33=-COSA*COSB
      AA34=0
      AA41=-Xfrom*COSA-Zfrom*SINA
      AA42=-Xfrom*SINA*SINB-Yfrom*COSB+Zfrom*COSA*SINB
      AA43=-Xfrom*SINA*COSB+Yfrom*SINB+Zfrom*COSA*COSB
      AA44=1
      RETURN
      END


      FUNCTION F(X1,Y1,X2,Y2,YSCAN)
      IF(ABS(YSCAN).GT.30000) THEN
        AYSCAN=30000
      ELSE
        AYSCAN=IFIX(YSCAN)
      ENDIF
      F=(AYSCAN-Y1)/(Y2-Y1)*(X2-X1)+X1
      RETURN
      END

      SUBROUTINE NORMAL(X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,A,B,C,D)
      A=(Y1-Y2)*(Z1+Z2)+(Y2-Y3)*(Z2+Z3)+(Y3-Y1)*(Z3+Z1)
      B=(Z1-Z2)*(X1+X2)+(Z2-Z3)*(X2+X3)+(Z3-Z1)*(X3+X1)
      C=(X1-X2)*(Y1+Y2)+(X2-X3)*(Y2+Y3)+(X3-X1)*(Y3+Y1)
      D=-(A*X1+B*Y1+C*Z1)
      RETURN
      END


      INTEGER FUNCTION IBRIGH(RND,Shine,IColor)
      IBRIGH=IColor
      IF (Shine.LT.15) Shine=15.
      IF (Shine.LT.(RND*60.)) IBRIGH=0
      IF (Shine.GT.(70.+RND*20.)) IBRIGH=7
      RETURN
      END
```

○アセンブラサブルーチン1

```
0000                    CODE      SEGMENT 'CODE'
                                  ASSUME CS:CODE
                        PUBLIC    CLS
0000                    CLS       PROC FAR
0000  FC                          CLD
0001  B8 A800                     MOV AX,0A800H
0004  8E C0                       MOV ES,AX
0006  B9 8000                     MOV CX,8000H
0009  BF 0000                     MOV DI,0
000C  B8 0000                     MOV AX,0
000F  F3/ AB                      REP STOSW
0011  B8 B800                     MOV AX,0B800H
0014  8E C0                       MOV ES,AX
0016  B9 4000                     MOV CX,4000H
0019  B8 0000                     MOV AX,0
001C  F3/ AB                      REP STOSW
001E  B5 C0                       MOV CH,0C0H
0020  B4 42                       MOV AH,42H
0022  CD 18                       INT 18H
0024  B4 40                       MOV AH,40H
0026  CD 18                       INT 18H
0028  CB                          DB 0CBH
0029                              CLS ENDP
0029                              CODE ENDS
                                  END
```

○アセンブラサブルーチン2

```
0000                    CODE      SEGMENT 'CODE'
                                  ASSUME CS:CODE
                        PUBLIC    PSET
0000                    PSET      PROC FAR
0000  55                          PUSH BP
0001  8B EC                       MOV BP,SP
0003  1E                          PUSH DS
0004  0E                          PUSH CS
0005  1F                          POP DS
0006  BB 003B R                   MOV BX,OFFSET ARGTBL
0009  C4 76 0E                    LES SI,[BP+14]
000C  26: 8B 04                   MOV AX,ES:[SI]
000F  89 47 08                    MOV [BX+8],AX
0012  C4 76 0A                    LES SI,[BP+10]
0015  26: 8B 04                   MOV AX,ES:[SI]
0018  89 47 0A                    MOV [BX+0AH],AX
001B  C4 76 06                    LES SI,[BP+6]
001E  26: 8B 04                   MOV AX,ES:[SI]
0021  88 07                       MOV [BX],AL
0023  C7 47 0C 0001               MOV WORD PTR [BX+0CH],1
0028  C7 47 0E 003A R             MOV WORD PTR [BX+0EH],OFFSET PATN
002D  0E                          PUSH CS
002E  07                          POP ES
002F  B5 B0                       MOV CH,0B0H
0031  B4 45                       MOV AH,45H
0033  CD 18                       INT 18H
0035  1F                          POP DS
0036  5D                          POP BP
0037  CA 0C 00                    DB 0CAH,0CH,0H
                        ;
003A  80                PATN      DB 80H
                        ;
003B  00 00 00 00 00 00 ARGTBL    DT 0
      00 00 00 00
0045  00 00 00 00 00 00           DT 0
      00 00 00 00
004F  00 00 00 00 00 00           DT 0
      00 00 00 00
0059  00 00 00 00 00 00           DT 0
      00 00 00 00
0063  00 00 00 00 00 00           DT 0
      00 00 00 00
006D                              PSET ENDP
006D                              CODE ENDS
                                  END
```

5．3　3次元地形図の入力・表示プログラム

5．3．1　はじめに

このプログラムは，本来，山脈のような形状を，三次元表示するためのものです．このような立体は，曲面で囲まれているわけですが，コンピュータにデータとして入力する場合には，ふつう数多くの平面で囲まれている図形として取りあつかう必要があります．このとき，立体を人手で平面に分割して，立体を平面の集まりとして入力しようとしますと(たとえば，平面の各頂点の座標を対にして)大変な手間になります．この手間を大はばに省くために，立体の頂点を入力して，頂点を結ぶ稜線を適当に計算した後，表示を行うプログラムを作製しました．

このプログラムは大きく分けて2つの部分から成りたっています．一つは，頂点の集合から稜線を引き，三角形の平面で立体を構成する部分，もう一方は，その立体を表示するための，三次元表示部分です．

では，各部のアルゴリズムを説明します．

5．3．2　稜線決定アルゴリズム

アルゴリズムには，1つの制限がつきます。それは，目的にはさしつかえがないのですが，立体の各面が1対1に$x-y$平面に投影するということです．すなわち，真上からすべての面が見えるような形になりますから，側面にくぼみがあったり，空間に浮かんだ立体には，適用できません．

この制限のために，アルゴリズムが簡単になり，プログラムもそれなりに早くなるのです．また，どうしても空間に浮かんだ凸多面体に応用する場合は，適当な$x-y$平面に平行な平面で，その立体を2分割してやれば，この方法が応用できます．

このアルゴリズムでは，各点のz軸の値を無視します．つまり，すべての点を$x-y$平面に投影した形で考えていきます．この状態で各点が三角形の頂点になるように，網状に分割すれば，各頂点の値も使って，三次元表示したときに，三角形で囲まれた矛盾のない立体になるのは，上記の制限によって明らかでしょう．

このプログラムは，2ステップで三角形分割を行ないます．まず，与えられた点をすべて含むような凸多角形をつくります．次にその多角形の内側へ次々に適当な三角形を決め，必要ならば新しい線を引く作業を行います．その過程で用いる主な表を表1に示します．PTBは点を管理する表です．プログラムに与える

点はこの形で指定しますが，前処理の段階としてソートして用います．ＬＴＢは，線を管理する表です．使用法は後に説明します．

I　外周を作る方法

与えられた点列をすべて含む，凸多角形の外周を作る方法は次のとおりです．

(1)　ＰＴＢをＸの小さい順にソートします．

(2)　ＰＴＢの最初の点をP_0 (X_0, Y_0) とします．

(3)　P_0から，ＰＴＢの下の方のすべての点について，線を引いたときの傾き$(Y-Y_0)/(X-X_0)$を求め，その最大値を与える点を見つけP_1とします．

(4)　P_0とP_1の間に線を引き，その線をＬＴＢに登録します．

(5)　P_1を新しいP_0として，(3)を繰り返します．

(6)　ＰＴＢの終りまできたら，ＰＴＢの終りの点をP_0として，ＰＴＢの上の方に向かって(3)～(5)を繰り返し，PTBの始めまできたら終りとします．

つまり，(1)～(5)の操作で，外周を半周分作り，(6)で逆向きに外周を作って，残りの半周を書き，外周を完成させるわけです．

ＬＴＢの登録法については，次の内部の三角形分割で説明します．

表1

PTB			
0	50	90	100
1	70	75	50
2	180	150	45
3	250	20	80
4			

●ＰＴＢはＸの値を用いて，使用前にソートします．

LTB	終点		頂点1	頂点2
0	1	3	NIL	2
1	3	4	NIL	INF
2	4	1	NIL	INF
3	1	2	3	INF
4	3	2	1	INF
5				

●点をＰＴＢのインデックス値で指定します．

●頂点１の値ＮＩＬは，その値をもつ線が外周の一部であることを示します．

●ＩＮＦは不定値，点が確定していないことを示します．

2 内部を三角形分割する方法

この部分は，本プログラムの中で最も複雑な部分です．三角形で分割した時に，新しい線分が他の線分と交叉してはいけないし，三角形でない部分（例えば四辺形）が残っても困ります．そのために，線分と三角形を，しっかり管理しなければなりません．そのためにあるのがＬＴＢです．

ＬＴＢは４つの部分からできています．始めの２つは線分の両端の点を指定するのに用います．残りの２つは，その線分が属する三角形のもう一つの項点を記録するのに用います．ふつう，１つの線分は，２つの三角形に属しています（図

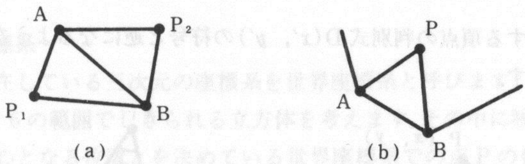


図１

1-a)から欄を２つ使いますが，線分が外周のときは，１つの三角形にしか属しません（図1-b)．そのため，第３番目の欄には，使わないことを表示するため，定数ＮＩＬを入れておきます．逆にいえば，第３番目の欄にＮＩＬがあれば，外周の線分であることがわかり，他の場合は，内部の線分であるとわかります．また，２つの三角形のうち１つは見つかっているが，もう１つは見つかっていないときは，第４番目の欄に，不定であることを示す定数ＩＮＦを入れておきます．この表の意味をよく頭において次のような手順を読んでください．

(1) ＬＴＢの上の方から，次々に線分に(2)以下の操作を加えます．
すでにこの段階では，2.1の部分で外周を形づくる線分が登録されていることに注意してください．また，ＬＴＢの第４欄がＩＮＦでなければ，その線分はいじる必要がないので，ここでぬかして次の線分を操作します．

(2) 操作を加える線分の第３欄がＮＩＬでしたら他のすべての点に対して，それ以外のときは，第３欄が示す項点の反対側にある点に対して，最良の三角形をつくる項点を捜します．

(3) (2)で求めた項点をＬＴＢの第４欄に登録します．そして求めた項点と線分の両端に線を引きますが，すでに作られた線分と共通になることがありますのでＬＴＢを調べ，もし共通線分でしたら，共通線分の第４欄に該当する項点を登録しますし，さもなければ新しい線分としてＬＴＢに登録します．

(4) ＬＴＢの最後の線分まで(2)，(3)の操作をひととおり行なえば終わりです．

簡単に書けばこれだけですが，特に(2)の項に鍵がいくつかかくされていますので少し説明を加えます．

まず，頂点を捜すときに，すべてにある頂点の反対側の点だけに注目することですが，これは図1-aを見ていただければおわかりのように2つの頂点は互いに他の反対側にあることから，必要性は明白だと思います．見分ける方法ですが，図-2のような状態のとき，判別式D

$$D\ (x,\ y) = (y - y_0) - \frac{y_0 - y_1}{x_0 - x_1}\ (x - x_0)$$

がすでに存在する頂点の判別式$D(x',\ y')$の符号と逆になるような点を見つければよいわけです．

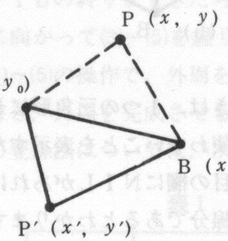


図2

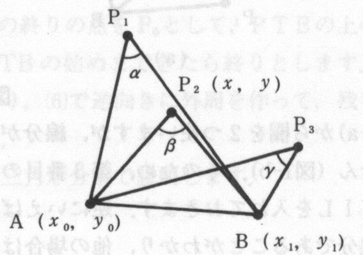


図3

次に適当な三角形を選ぶ方法について述べます．すでにおわかりのことと思いますが，この三角形の中に他の点があったり，三角形の辺が他の線と交わらないようにしなければなりません．そこで，たとえば図-3のような状態のとき，どの三角形を選ぶかという問題になりますが，今回採用したアルゴリズムは，三角形の頂角を最大にするものを選ぶということにしました．すなわち図-3では，角βを与える点P_2を選ぶということです．ただし，角βを直接求めたのでは手間がかかりますので，角θが$0<\theta<\pi$の区間で単調減少する関数，$\cos(\theta)$を用いることにしました．$\cos(\theta)$を求めるには，

$$\cos(\theta) = \frac{(x - x_0)\ (x - x_1) + (y - y_0)\ (y - y_1)}{\sqrt{(x - x_0)^2 + (y - y_0)^2}\ \sqrt{(x - x_1)^2 + (y - y_1)^2}}$$

を用い，$\cos(\theta)$を最小にする点$P(x,\ y)$を捜せばよいことになります．この方法でまちがいなく上記の諸条件を満足していることを確かめてください．

今までの説明では,「線を引く」というと, 画面上の$x-y$平面にそのまま描画しているような感じがしましたが, 実際は表示の部分は, z 軸方向の高さも計算に入れて3次元描画しています. その方法については, 次の節をごらんください.

5・3・3　座標変換アルゴリズム

三次元の空間に存在している物体を二次元の平面で表現する時には座標系の変換を行ないます. はじめに, 今回変換に使用しました座標系について説明します.

1　世界座標系

物体が存在している三次元の座標系を世界座標系と呼びます. X, Y, Z それぞれ, ±4096の範囲でしきられる立方体を考えます. その中に物体と参照点つまり投影の中心となる位置とを決めている世界座標系での点Pの位置はP (X, Y, Z) で表わします. 実際には, 図4のように右手系の三次元直交座標系を用います. そこで, この座標系での参照点Eは, (X_0, Y_0, Z_0) で示されている位置に存在しているとします.

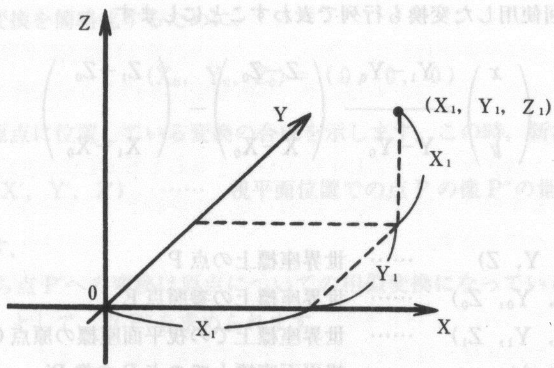


図4

2　視平面座標系

参照点について物体と同じ側に, ある平面をおいた時に物体の像をその平面に投影することができます. それが実際に二次元で表示する時の像になり, ディスプレイや$x-y$プロッタのような平面にも, 三次元のデータを持つ物体の像などを表示することができるわけです. この座標系で点P′は, P′ (x, y) で表わします. 実際の座標を, 図5のように二次元直交座標系にとります. しかし, 私達が

普段見なれているy軸の方向と逆を向いていますが，これは本書の３章でも示したように，ＰＣ-９８０１のグラフィックＬＩＯの座標系に合わせたためです．したがって原点は，x，yを正の値に限った時には，平面の左上すみになります．

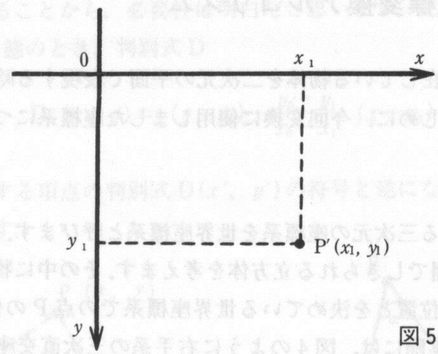


図５

3　次元の変換例

座標の変換には，行列を使って解く方法が便利でかつ一般に広く用いられていますので，今回使用した変換も行列で表わすことにします．

$$\begin{pmatrix} x \\ y \end{pmatrix} = \frac{Y_1 - Y_0}{Y - Y_0} \begin{pmatrix} Z - Z_0 \\ X - X_0 \end{pmatrix} - \begin{pmatrix} Z_1 - Z_0 \\ X_1 - X_0 \end{pmatrix}$$

ここで，

(X，Y，Z)　……　世界座標上の点Ｐ
(X_0，Y_0，Z_0)　……　世界座標上の参照点Ｅ
(X_1，Y_1，Z_1)　……　世界座標上での視平面座標の原点Ｏ
(x，y)　……　視平面座標上での点Ｐの像P′

です．以上の位置関係を　Ｙ－Ｚ平面に投影した見取り図を図６に示します．

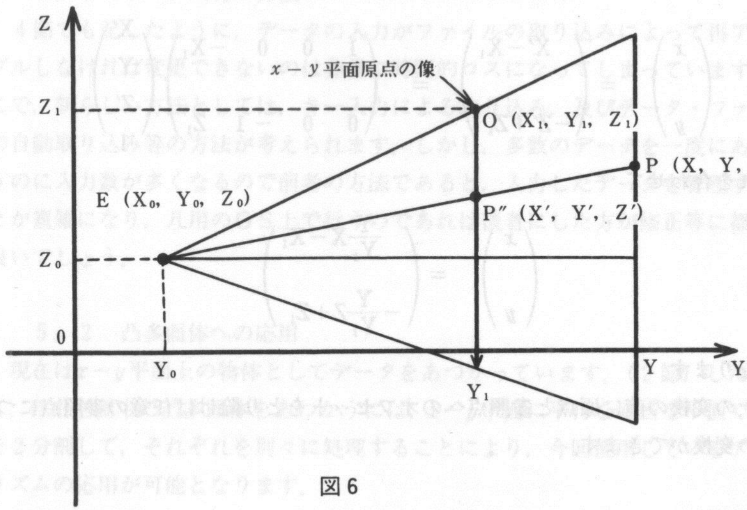


図6

今回の試作では視平面の方向は，X−Z 平面に常に平行にたもたれています．

上記の変換を簡略化するために，

$$(X_0, Y_0, Z_0) = (0, 0, 0)$$

すなわち原点に位置している変換の合成を示します．この時，新たな変数として

(X', Y', Z') …… 視平面位置での点 P の像 P″の世界座標

を考えます．

点 P から点 P′への変換は原点についての相似変換になっているので，倍率を (Y/Y_1) として，点 P″を求められます．

$$\begin{pmatrix} X' \\ Y' \\ Z' \\ 1 \end{pmatrix} = \begin{pmatrix} Y/Y_1 & 0 & 0 & 0 \\ 0 & Y/Y_1 & 0 & 0 \\ 0 & 0 & Y/Y_1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

次に視平面上においての変換は

$$\begin{pmatrix} x \\ \\ y \end{pmatrix} = \begin{pmatrix} X'-X_1 \\ \\ -Z'+Z_1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & -X_1 \\ \\ 0 & 0 & -1 & Z_1 \end{pmatrix} \begin{pmatrix} X' \\ Y' \\ Z' \\ 1 \end{pmatrix}$$

これを合わせて

$$\begin{pmatrix} x \\ \\ y \end{pmatrix} = \begin{pmatrix} \frac{Y}{Y_1}X-X_1 \\ \\ -\frac{Y}{Y_1}Z+Z_1 \end{pmatrix}$$

となります．

以上の変換の前に原点と参照点へのオフセットをとり除けば任意の参照点についての変換ができます．

4　使用法

対象となるデータは，外部のデータ・ファイル（TABLE．DAT）で定義されているものであり，CP/M-86でのASM86の疑似命令INCLUDEでとり込まれますので，新しいデータを使用する時は，2節及びリスト2を参考にして，ファイルを書き直して下さい．ポイントの数（PMAX＋1）は256個まで可能です．また，HDVが1のとき，高解像度，0のとき低解像度モニタで動作します．

頂点のプロットはグラフィックLIOのGCIRCLEとGPSETの組み合わせになっていますので，円の半径，円周の色，中央のドットの色などを簡単に変化させることができます．

頂点間のラインの形式等は一定にしてありますがパラメータの変更で可能にすることができます．（GLINE参照）

CP/M-86でのメモリはスモール・モデルで管理されています．プログラム前半及びストップ・キーによる割込みなどの割込みルーチンのテーブルを保存するようになっているので，他のプログラムなどからの割込みルーチンとしての応用も可能です．

5　将来への改良点

現段階のプログラムは機能的にまだ不十分な点を残しています．そのいくつかを次に記します．

5．1 データ入力の方法

4節でも記したように，データの入力がファイルの取り込みによって再アセンブルしなければ変更できないのは非常に時間的ロスになってしまっています．そこで，新らしい方法としては，キー入力による取り込み，及びデータ・ファイルの自動取り込み等の方法が考えられます．しかし，多数のデータを一度にあつかうのに入力数が多くなるので前者の方法であると，入力したデータを管理することが複雑になり，凡用のＯＳ上で行うのであれば後者にした方が修正等に都合が良いでしょう．

5．2 凸多面体への応用

現在は$x-y$平面上の物体としてデータをあつかっています．（2節）したがって，自由空間中の凸多面体をあつかうには，$x-y$平面に平行な適当な平面で物体を2分割して，それぞれを別々に処理することにより，今回使用しているアルゴリズムの応用が可能となります．

5．3 等高線を物体に書き込む

ワイヤー・フレームの各線に対してある一定の高さの点をマークし，それを結んで行けば一本の等高線を引くことができます．そこで適当な間隔で等高線を引いて行けば等高線表示を行なえます．よって，より物体の形状をとらえることができますが，峰をいくつか持っているような物体に関しては実際につながってはならない等高線がつながってしまう場合がありえるので，その処理については注意しなければなりません。

5．4 陰線処理

現在のバージョンでは陰線処理が行なわれていませんが，これを行なうことにより，より一層見やすくなります．

5．5 シェイディング ：Shading

5．4にさらにシェイディング（陰影付け）を行なうことにより，さらに立体感が増し，リアルな図形を表現できます．

5．6 スムージング ：Smoothing

今まで図形を多面体近似してましたが，実際の図形は丸みを持っていることが多いのです．したがって，より高度な応用になりますが今回作成した図形にスム

ージングをほどこして表面をなめらかにするのも良いでしょう．ただし現在のマイクロコンピューターではかなりむずかしい問題となるでしょう．

5．7 座標変換の多様性

今回作成した変換パッケージは基本的な部分しかサポートされていません．そこで標準三次元パッケージと同様なパッケージを作ることによって，図形表現に変化をもたせ，ハードウェアに依存しないグラフィックを行なうことも可能です．

リスト1

```
                         ;=================================================================
                         ;=== 3D WIRE FRAME MODEL DISPLAY SYSTEM                  ========
                         ;===      ON PC-9800 GRAPHIC LIO & CP/M-86               ========
                         ;===                                                     ========
                         ;===                                                     ========
                         ;=================================================================
                                      TITLE  '3D WIRE FRAME MODEL DISPLAY SYSTEM'
 00A0                    GINIT   EQU      0A0H
 00A1                    GSCREEN EQU      0A1H
 00A2                    GVIEW   EQU      0A2H
 00A3                    GCOLOR1 EQU      0A3H
 00A4                    GCOLOR2 EQU      0A4H
 00A5                    GCLS    EQU      0A5H
 00A6                    GPSET   EQU      0A6H
 00A7                    GLINE   EQU      0A7H
 00A8                    GCIRCLE EQU      0A8H
 00A9                    GPAINT1 EQU      0A9H
 00AA                    GPAINT2 EQU      0AAH
 00AB                    GGET    EQU      0ABH
 00AC                    GPUT1   EQU      0ACH
 00AD                    GPUT2   EQU      0ADH
 00AE                    GROLL   EQU      0AEH
 00AF                    GPOINT2 EQU      0AFH
 00CE                    GCOPY   EQU      0CEH
                         ;
 00E0                    BDOS    EQU      224

                                      CSEG

                         ;=============================================
                         ;===                                      ===
                         ;=== INTERRUPT VECTOR INITIALISE ==========
                         ;===                                      ===
                         ;=============================================
                         ;=======================================
                         ;=== SET STACK AREA ====================
                         ;=======================================
 0000 9C                              PUSHF
 0001 5B                              POP     BX
 0002 FA                              CLI
 0003 8CD8                            MOV     AX,DS
 0005 8ED0                            MOV     SS,AX
 0007 BC5E28                          MOV     SP,OFFSET STACK_TOP
 000A 53                              PUSH    BX
 000B 9D                              POPF
                         ;=======================================
                         ;=== STORE DATA SEGMENT ADDRES ====
                         ;=======================================

 000C 2E8C1EF604                      MOV     CS:DS_BACKUP,DS
 0011 2E8C16F804                      MOV     CS:SS_BACKUP,SS
 0016 2E8926FA04                      MOV     CS:SP_BACKUP,SP
                         ;=======================================
                         ;=== SAVE INTERRUPT VECTORS =======
                         ;=======================================
                         ;
```

```
                         ;=== GRAPHIC LIO =================
001B FA                        CLI
001C 1E                        PUSH    DS
001D 07                        POP     ES
001E 33C0                      XOR     AX,AX
0020 8ED8                      MOV     DS,AX
0022 BE8002                    MOV     SI,0A0H*4
0025 BF0001                    MOV     DI,OFFSET VECT_BACKUP
0028 B92000                    MOV     CX,32
002B FC                        CLD
002C F3A5                      REP     MOVSW
002E BE3803                    MOV     SI,0CEH*4
0031 A5                        MOVSW
0032 A5                        MOVSW
                         ;=== STOP CHECK VECTOR ===
0033 BE1403                    MOV     SI,0C5H*4
0036 A5                        MOVSW
0037 A5                        MOVSW
                         ;=== STOP KEY VECTOR ===
0038 BE1800                    MOV     SI,6*4
003B A5                        MOVSW
003C A5                        MOVSW
                         ;==========================================
                         ;===                                   ===
                         ;=== SET INTERRUPT VECTOR ================
                         ;===                                   ===
                         ;==========================================
                         ;
                         ;=== GRAPHIC LIO =================
                         ;
003D 06                        PUSH    ES
003E B91000                    MOV     CX,16
0041 B890F9                    MOV     AX,LIO_BASE
0044 8ED8                      MOV     DS,AX
0046 33C0                      XOR     AX,AX
0048 8EC0                      MOV     ES,AX
004A BE0600                    MOV     SI,6
004D BF8002                    MOV     DI,0A0H*4
                 LP_SETV:
0050 A5                        MOVSW
0051 B890F9                    MOV     AX,LIO_BASE
0054 AB                        STOSW
0055 83C602                    ADD     SI,2
0058 E2F6           0050       LOOP    LP_SETV
005A BF3803                    MOV     DI,0CEH*4
005D A5                        MOVSW
005E B890F9                    MOV     AX,LIO_BASE
0061 AB                        STOSW
                         ;=== CHECK STOP =================
0062 BF1403                    MOV     DI,0C5H*4
0065 B8BF04                    MOV     AX,OFFSET CHECK_STOP
0068 AB                        STOSW
0069 268C0D                    MOV     ES:[DI],CS
                         ;=== STOP KEY ===================
006C BF1800                    MOV     DI,6*4
006F B8B204                    MOV     AX,OFFSET STOP_KEY
0072 AB                        STOSW
0073 268C0D                    MOV     ES:[DI],CS
0076 1F                        POP     DS
0077 FB                        STI
                         ;
                         ;
 7FFF                    MAXVAL  EQU     32767
 8000                    MINVAL  EQU     -32768
 FFFE                    NIL     EQU     -2
 FFFF                    INF     EQU     -1
                         ;
                         ;==========================================
                         ;===                                   ===
                         ;=== MAIN ROUTINE ========================
                         ;===                                   ===
                         ;==========================================
                         ;
0078 B80000              MAIN:   MOV     AX,0
007B A35620                      MOV     LP,AX
007E E88600         0107         CALL    PSORT
                         ;
0081 E8AE03         0432         CALL    TRN_3D2D
                         ;
```

```
0084 B500                          MOV     CH,0
0086 BA0080             OLIN1:     MOV     DX,MINVAL
0089 8ACD                          MOV     CL,CH
008B 3A0E4F1C           OLIN2:     CMP     CL,BYTE PTR PMAX
008F 7407         0098             JE      OLIN3
0091 FEC1                          INC     CL
0093 E8B100       0147             CALL    CKLIN1
0096 EBF3         008B             JMPS    OLIN2
0098 E8EA00       0185  OLIN3:     CALL    MKLIN1
009B 3A1E4F1C                      CMP     BL,BYTE PTR PMAX
009F 7404         00A5             JE      OLIN4
00A1 8AEB                          MOV     CH,BL
00A3 EBE1         0086             JMPS    OLIN1
                        ;
00A5 8A2E4F1C           OLIN4:     MOV     CH,BYTE PTR PMAX
00A9 BA0080             OLIN5:     MOV     DX,MINVAL
00AC 8ACD                          MOV     CL,CH
00AE FEC9               OLIN6:     DEC     CL
00B0 7805         00B7             JS      OLIN7
00B2 E89200       0147             CALL    CKLIN1
00B5 EBF7         00AE             JMPS    OLIN6
00B7 E8CB00       0185  OLIN7:     CALL    MKLIN1
00BA 80FB00                        CMP     BL,0
00BD 7404         00C3             JE      ILIN
00BF 8AEB                          MOV     CH,BL
00C1 EBE6         00A9             JMPS    OLIN5
                        ;
00C3 B80000             ILIN:      MOV     AX,0
00C6 A35820                        MOV     LS,AX
00C9 A15820             ILIN1:     MOV     AX,LS
00CC 3B065620                      CMP     AX,LP
00D0 7503         00D5             JNE     DUMM
00D2 E9D703       04AC             JMP     REBOOT
00D5 03C0               DUMM:      ADD     AX,AX
00D7 03C0                          ADD     AX,AX
00D9 8BF0                          MOV     SI,AX
00DB 8A9C591C                      MOV     BL,[SI]+OFFSET LTB+3
00DF 80FBFF                        CMP     BL,INF
00E2 7406         00EA             JE      ILIN2
00E4 FF065820                      INC     LS
00E8 EBDF         00C9             JMPS    ILIN1
00EA E88602       0373  ILIN2:     CALL    MKTRI
00ED 8884591C                      MOV     [SI]+OFFSET LTB+3,AL
00F1 8AAC561C                      MOV     CH,[SI]+OFFSET LTB
00F5 8A8C571C                      MOV     CL,[SI]+OFFSET LTB+1
00F9 E8F401       02F0             CALL    REGLN
00FC 86E9                          XCHG    CH,CL
00FE E8EF01       02F0             CALL    REGLN
0101 FF065820                      INC     LS
0105 EBC2         00C9             JMPS    ILIN1
                        ;
                        ;
                        ;==================================
                        ;===                            ===
                        ;=== PTB SORTING ==================
                        ;===                            ===
                        ;==================================
                        ;
0107 BB0D1C             PSORT:     MOV     BX,OFFSET PTB
010A B500                          MOV     CH,0
010C BAFF7F             PSORT1:    MOV     DX,MAXVAL
010F 8ACD                          MOV     CL,CH
0111 BE0000                        MOV     SI,0
0114 3B10               PSORT2:    CMP     DX,[BX+SI]
0116 7E04         011C             JLE     PSORT3
0118 8B10                          MOV     DX,[BX+SI]
011A 8BFE                          MOV     DI,SI
011C 3A0E4F1C           PSORT3:    CMP     CL,BYTE PTR PMAX
0120 7407         0129             JE      PSORT4
0122 FEC1                          INC     CL
0124 83C606                        ADD     SI,6
0127 EBEB         0114             JMPS    PSORT2
0129 BE0000             PSORT4:    MOV     SI,0
012C E80F00       013E             CALL    PSORT5
012F E80C00       013E             CALL    PSORT5
0132 E80900       013E             CALL    PSORT5
0135 FEC5                          INC     CH
0137 3A2E4F1C                      CMP     CH,BYTE PTR PMAX
```

```
013B 75CF      010C          JNE     PSORT1
013D C3                      RET
                     ;
013E 8B00            PSORT5: MOV     AX,[BX+SI]
0140 8701                    XCHG    AX,[BX+DI]
0142 8900                    MOV     [BX+SI],AX
0144 43                      INC     BX
0145 43                      INC     BX
0146 C3                      RET
                     ;
                     ;
                     ;=================================
                     ;===                           ===
                     ;=== CHECK OUTER LINE ============
                     ;===                           ===
                     ;=================================
                     ;
0147 53              CKLIN1: PUSH    BX
0148 52                      PUSH    DX
0149 B400                    MOV     AH,0
014B 8AC5                    MOV     AL,CH
014D E86200    01B2          CALL    GETPTB
0150 8BD8                    MOV     BX,AX
0152 B400                    MOV     AH,0
0154 8AC1                    MOV     AL,CL
0156 E85900    01B2          CALL    GETPTB
0159 2BC3                    SUB     AX,BX
015B 7425      0182          JZ      CKLNR1
015D 8BD8                    MOV     BX,AX
                     ;
015F B401                    MOV     AH,1
0161 8AC5                    MOV     AL,CH
0163 E84C00    01B2          CALL    GETPTB
0166 8BD0                    MOV     DX,AX
0168 B401                    MOV     AH,1
016A 8AC1                    MOV     AL,CL
016C E84300    01B2          CALL    GETPTB
016F 2BC2                    SUB     AX,DX
0171 F72E521C                IMUL    WORD PTR NUM8
0175 F7FB                    IDIV    BX
0177 5A                      POP     DX
0178 5B                      POP     BX
0179 3BC2                    CMP     AX,DX
017B 7E04      0181          JLE     CKLNR2
017D 8BD0                    MOV     DX,AX
017F 8AD9                    MOV     BL,CL
0181 C3              CKLNR2: RET
                     ;
0182 5A              CKLNR1: POP     DX
0183 5B                      POP     BX
0184 C3                      RET
                     ;
                     ;
                     ;=================================
                     ;===                           ===
                     ;=== MAKE OUTER LINE =============
                     ;===                           ===
                     ;=================================
                     ;
0185 57              MKLIN1: PUSH    DI
0186 8AC5                    MOV     AL,CH
0188 8AE3                    MOV     AH,BL
018A B207                    MOV     DL,7
018C E8BB03    054A          CALL    LINE
018F A15620                  MOV     AX,LP
0192 D1E0                    SAL     AX,1
0194 D1E0                    SAL     AX,1
0196 8BF8                    MOV     DI,AX
0198 88AD561C                MOV     [DI]+OFFSET LTB,CH
019C 889D571C                MOV     [DI]+OFFSET LTB+1,BL
01A0 B0FE                    MOV     AL,NIL
01A2 8885581C                MOV     [DI]+OFFSET LTB+2,AL
01A6 B0FF                    MOV     AL,INF
01A8 8885591C                MOV     [DI]+OFFSET LTB+3,AL
01AC FF065620                INC     LP
01B0 5F                      POP     DI
01B1 C3                      RET
                     ;
                     ;
```

```
                         ;==================================
                         ;===                            ===
                         ;=== GET DATUM FROM PTB ==========
                         ;===                            ===
                         ;==================================
                         ;
01B2 56                  GETPTB: PUSH   SI
01B3 52                          PUSH   DX
01B4 8AD4                        MOV    DL,AH
01B6 B600                        MOV    DH,0
01B8 F626511C                    MUL    BYTE PTR NUM6
01BC 03C2                        ADD    AX,DX
01BE 03C2                        ADD    AX,DX
01C0 8BF0                        MOV    SI,AX
01C2 8B840D1C                    MOV    AX,[SI]+OFFSET PTB
01C6 5A                          POP    DX
01C7 5E                          POP    SI
01C8 C3                          RET
                         ;
                         ;==================================
                         ;===                            ===
                         ;=== SQUARE ROOT AX ==============
                         ;===                            ===
                         ;==================================
                         ;
01C9 53                  SQRT:   PUSH   BX
01CA 51                          PUSH   CX
01CB 8BD8                        MOV    BX,AX
01CD B80100                      MOV    AX,1
01D0 8BC8                        MOV    CX,AX
01D2 3BC3                SQRT1:  CMP    AX,BX
01D4 7D08         01DE           JGE    SQRT2
01D6 03C1                        ADD    AX,CX
01D8 03C1                        ADD    AX,CX
01DA 40                          INC    AX
01DB 41                          INC    CX
01DC EBF4         01D2           JMPS   SQRT1
01DE 8BC1                SQRT2:  MOV    AX,CX
01E0 59                          POP    CX
01E1 5B                          POP    BX
01E2 C3                          RET
                         ;
                         ;==================================
                         ;===                            ===
                         ;=== VECTOR LENGTH ===============
                         ;===                            ===
                         ;==================================
                         ;
01E3 53                  FNDT:   PUSH   BX
01E4 52                          PUSH   DX
01E5 B400                        MOV    AH,0
01E7 8AC5                        MOV    AL,CH
01E9 E8C6FF       01B2           CALL   GETPTB
01EC 8BD8                        MOV    BX,AX
01EE B400                        MOV    AH,0
01F0 8AC1                        MOV    AL,CL
01F2 E8BDFF       01B2           CALL   GETPTB
01F5 2BC3                        SUB    AX,BX
01F7 F7E8                        IMUL   AX
01F9 F736541C                    DIV    NUM512
01FD 50                          PUSH   AX
                         ;
01FE B401                        MOV    AH,1
0200 8AC5                        MOV    AL,CH
0202 E8ADFF       01B2           CALL   GETPTB
0205 8BD8                        MOV    BX,AX
0207 B401                        MOV    AH,1
0209 8AC5                        MOV    AL,CH
020B E8A4FF       01B2           CALL   GETPTB
020E 2BC3                        SUB    AX,BX
0210 F7E8                        IMUL   AX
0212 F736541C                    DIV    NUM512
0216 5B                          POP    BX
0217 03C3                        ADD    AX,BX
0219 5A                          POP    DX
021A 5B                          POP    BX
021B C3                          RET
                         ;
```

```
                                ;
                                ;=================================
                                ;===                          ===
                                ;=== VECTOR DOT PRODUCT ===========
                                ;===                          ===
                                ;=================================
                                ;
021C 53                         FNARC:  PUSH    BX
021D A25A20                             MOV     FNXB0,AL
0220 B400                               MOV     AH,0
0222 8AC5                               MOV     AL,CH
0224 E88BFF         01B2                CALL    GETPTB
0227 8BD8                               MOV     BX,AX
0229 B400                               MOV     AH,0
022B 8AC1                               MOV     AL,CL
022D E882FF         01B2                CALL    GETPTB
0230 50                                 PUSH    AX
0231 2BC3                               SUB     AX,BX
0233 8BD8                               MOV     BX,AX
0235 B400                               MOV     AH,0
0237 A05A20                             MOV     AL,FNXB0
023A E875FF         01B2                CALL    GETPTB
023D 8BD0                               MOV     DX,AX
023F 58                                 POP     AX
0240 2BC2                               SUB     AX,DX
0242 F7EB                               IMUL    BX
0244 52                                 PUSH    DX
0245 50                                 PUSH    AX
0246 B401                               MOV     AH,1
0248 8AC5                               MOV     AL,CH
024A E865FF         01B2                CALL    GETPTB
024D 8BD8                               MOV     BX,AX
024F B401                               MOV     AH,1
0251 8AC1                               MOV     AL,CL
0253 E85CFF         01B2                CALL    GETPTB
0256 50                                 PUSH    AX
0257 2BC3                               SUB     AX,BX
0259 8BD8                               MOV     BX,AX
025B B401                               MOV     AH,1
025D A05A20                             MOV     AL,FNXB0
0260 E84FFF         01B2                CALL    GETPTB
0263 8BD0                               MOV     DX,AX
0265 58                                 POP     AX
0266 2BC2                               SUB     AX,DX
0268 F7EB                               IMUL    BX
026A 5B                                 POP     BX
026B 03C3                               ADD     AX,BX
026D 5B                                 POP     BX
026E 13D3                               ADC     DX,BX
0270 5B                                 POP     BX
0271 C3                                 RET
                                ;
                                ;=================================
                                ;===                          ===
                                ;=== DESCLIMINATE FUNCTION ========
                                ;===                          ===
                                ;=================================
                                ;
0272 53                         FNX:    PUSH    BX
0273 52                                 PUSH    DX
0274 56                                 PUSH    SI
0275 A15820                             MOV     AX,LS
0278 03C0                               ADD     AX,AX
027A 03C0                               ADD     AX,AX
027C 8BF0                               MOV     SI,AX
027E 8A84561C                           MOV     AL,[SI]+OFFSET LTB
0282 A25A20                             MOV     FNXB0,AL
0285 8A84571C                           MOV     AL,[SI]+OFFSET LTB+1
0289 A25B20                             MOV     FNXB1,AL
                                ;
028C B401                               MOV     AH,1
028E 8AC1                               MOV     AL,CL
0290 E81FFF         01B2                CALL    GETPTB
0293 50                                 PUSH    AX
0294 B401                               MOV     AH,1
0296 A05A20                             MOV     AL,FNXB0
0299 E816FF         01B2                CALL    GETPTB
029C 8BD8                               MOV     BX,AX
```

```
029E 58                              POP    AX
029F 2BC3                            SUB    AX,BX
02A1 50                              PUSH   AX
02A2 B401                            MOV    AH,1
02A4 A05A20                          MOV    AL,FNXB0
02A7 E808FF         01B2             CALL   GETPTB
02AA 8BD8                            MOV    BX,AX
02AC B401                            MOV    AH,1
02AE A05B20                          MOV    AL,FNXB1
02B1 E8FEFE         01B2             CALL   GETPTB
02B4 2BC3                            SUB    AX,BX
02B6 8BD0                            MOV    DX,AX
02B8 B400                            MOV    AH,0
02BA A05A20                          MOV    AL,FNXB0
02BD E8F2FE         01B2             CALL   GETPTB
02C0 8BD8                            MOV    BX,AX
02C2 B400                            MOV    AH,0
02C4 8AC1                            MOV    AL,CL
02C6 E8E9FE         01B2             CALL   GETPTB
02C9 2BC3                            SUB    AX,BX
02CB F7EA                            IMUL   DX
02CD 50                              PUSH   AX
02CE B400                            MOV    AH,0
02D0 A05A20                          MOV    AL,FNXB0
02D3 E8DCFE         01B2             CALL   GETPTB
02D6 8BD8                            MOV    BX,AX
02D8 B400                            MOV    AH,0
02DA A05B20                          MOV    AL,FNXB1
02DD E8D2FE         01B2             CALL   GETPTB
02E0 2BC3                            SUB    AX,BX
02E2 8BD8                            MOV    BX,AX
02E4 58                              POP    AX
02E5 F7FB                            IDIV   BX
02E7 8BD8                            MOV    BX,AX
02E9 58                              POP    AX
02EA 2BC3                            SUB    AX,BX
02EC 5E                              POP    SI
02ED 5A                              POP    DX
02EE 5B                              POP    BX
02EF C3                              RET
                            ;
                            ;
                            ;==================================
                            ;===                            ===
                            ;=== REGIST LINE ==================
                            ;===                            ===
                            ;==================================
                            ;
02F0 56                     REGLN:   PUSH   SI
02F1 E83D00         0331             CALL   SRCHLN
02F4 80FCFF                          CMP    AH,-1
02F7 7412           030B             JE     REGLN1
02F9 50                              PUSH   AX
02FA 8AC4                            MOV    AL,AH
02FC B400                            MOV    AH,0
02FE 03C0                            ADD    AX,AX
0300 03C0                            ADD    AX,AX
0302 8BF0                            MOV    SI,AX
0304 58                              POP    AX
0305 888C591C                        MOV    [SI]+OFFSET LTB+3,CL
0309 5E                              POP    SI
030A C3                              RET
                            ;
030B 8B365620               REGLN1:  MOV    SI,LP
030F 03F6                            ADD    SI,SI
0311 03F6                            ADD    SI,SI
0313 8884561C                        MOV    [SI]+OFFSET LTB,AL
0317 88AC571C                        MOV    [SI]+OFFSET LTB+1,CH
031B 888C581C                        MOV    [SI]+OFFSET LTB+2,CL
031F C684591CFF                      MOV    BYTE PTR [SI]+OFFSET LTB+3,INF
0324 FF065620                        INC    LP
0328 8AE5                            MOV    AH,CH
032A B207                            MOV    DL,7
032C E81B02         054A             CALL   LINE
032F 5E                              POP    SI
0330 C3                              RET
                            ;
                            ;
                            ;==================================
```

```
                               ;===                                 ===
                               ;=== SERCH LINE ======================
                               ;===                                 ===
                               ;=====================================
                               ;
0331 53                        SRCHLN: PUSH    BX
0332 BB0000                            MOV     BX,0
0335 8BF3                              MOV     SI,BX
0337 3A84561C                  SCHLN1: CMP     AL,[SI]+OFFSET LTB
033B 7506             0343             JNE     SCHLN2
033D 3AAC571C                          CMP     CH,[SI]+OFFSET LTB+1
0341 7429             036C             JE      SCHLNR
0343 43                        SCHLN2: INC     BX
0344 83C604                            ADD     SI,4
0347 3B1E5620                          CMP     BX,LP
034B 75EA             0337             JNE     SCHLN1
                               ;
034D BB0000                            MOV     BX,0
0350 8BF3                              MOV     SI,BX
0352 3AAC561C                  SCHLN3: CMP     CH,[SI]+OFFSET LTB
0356 7506             035E             JNE     SCHLN4
0358 3A84571C                          CMP     AL,[SI]+OFFSET LTB+1
035C 740E             036C             JE      SCHLNR
035E 43                        SCHLN4: INC     BX
035F 83C604                            ADD     SI,4
0362 3B1E5620                          CMP     BX,LP
0366 75EA             0352             JNE     SCHLN3
0368 5B                                POP     BX
0369 B4FF                              MOV     AH,-1
036B C3                                RET
                               ;
036C 8AE3                      SCHLNR: MOV     AH,BL
036E 5B                                POP     BX
036F C3                                RET
                               ;
                               ;
                               ;=====================================
                               ;===                                 ===
                               ;=== MAKE TRIANGLE ===================
                               ;===                                 ===
                               ;=====================================
0370 E9B800           042B MKTRR1:     JMP     MKTRR
                               ;
0373 53                        MKTRI:  PUSH    BX
0374 51                                PUSH    CX
0375 52                                PUSH    DX
0376 56                                PUSH    SI
0377 8B365820                          MOV     SI,LS
037B 03F6                              ADD     SI,SI
037D 03F6                              ADD     SI,SI
037F 8A84581C                          MOV     AL,[SI]+OFFSET LTB+2
0383 B300                              MOV     BL,0
0385 3CFE                              CMP     AL,NIL
0387 740D             0396             JE      MKTR1
0389 B301                              MOV     BL,1
038B 8AC8                              MOV     CL,AL
038D E8E2FE           0272             CALL    FNX
0390 23C0                              AND     AX,AX
0392 7802             0396             JS      MKTR1
0394 B302                              MOV     BL,2
0396 881E5C20                  MKTR1:  MOV     MD,BL
039A BAFF7F                            MOV     DX,MAXVAL
039D B300                              MOV     BL,0
039F FECB                              DEC     BL
03A1 E94B00           03EF             JMP     MKTR3
03A4 3A9C561C                  MKTR2:  CMP     BL,[SI]+OFFSET LTB
03A8 7445             03EF             JE      MKTR3
03AA 3A9C571C                          CMP     BL,[SI]+OFFSET LTB+1
03AE 743F             03EF             JE      MKTR3
03B0 52                                PUSH    DX
03B1 8AEB                              MOV     CH,BL
03B3 8A8C561C                          MOV     CL,[SI]+OFFSET LTB
03B7 E829FE           01E3             CALL    FNDT
03BA E80CFE           01C9             CALL    SQRT
03BD 8BD0                              MOV     DX,AX
03BF 8AEB                              MOV     CH,BL
03C1 8A8C571C                          MOV     CL,[SI]+OFFSET LTB+1
03C5 E81BFE           01E3             CALL    FNDT
03C8 E8FEFD           01C9             CALL    SQRT
```

```
03CB F7E2                            MUL     DX
03CD 03C0                            ADD     AX,AX
03CF 03C0                            ADD     AX,AX
03D1 03C0                            ADD     AX,AX
03D3 8ACB                            MOV     CL,BL
03D5 53                              PUSH    BX
03D6 8BD8                            MOV     BX,AX
03D8 8AAC561C                        MOV     CH,[SI]+OFFSET LTB
03DC 8A84571C                        MOV     AL,[SI]+OFFSET LTB+1
03E0 E839FE       021C               CALL    FNARC
03E3 F7FB                            IDIV    BX
03E5 5B                              POP     BX
03E6 5A                              POP     DX
03E7 3BC2                            CMP     AX,DX
03E9 7D04         03EF               JGE     MKTR3
03EB 8BD0                            MOV     DX,AX
03ED 8AFB                            MOV     BH,BL
03EF A05C20              MKTR3:      MOV     AL,MD
03F2 3C00                            CMP     AL,0
03F4 750A         0400               JNE     MKTR4
03F6 FEC3                            INC     BL
03F8 3A1E4F1C                        CMP     BL,PMAX
03FC 742D         042B               JE      MKTRR
03FE EBA4         03A4               JMPS    MKTR2
0400 3C01                MKTR4:      CMP     AL,1
0402 7413         0417               JE      MKTR6
0404 FEC3                MKTR5:      INC     BL
0406 3A1E4F1C                        CMP     BL,PMAX
040A 741F         042B               JE      MKTRR
040C 8ACB                            MOV     CL,BL
040E E861FE       0272               CALL    FNX
0411 23C0                            AND     AX,AX
0413 78EF         0404               JS      MKTR5
0415 EB8D         03A4               JMPS    MKTR2
0417 FEC3                MKTR6:      INC     BL
0419 3A1E4F1C                        CMP     BL,PMAX
041D 740C         042B               JE      MKTRR
041F 8ACB                            MOV     CL,BL
0421 E84EFE       0272               CALL    FNX
0424 23C0                            AND     AX,AX
0426 79EF         0417               JNS     MKTR6
0428 E979FF       03A4               JMP     MKTR2
                         ;
042B 8AC7                MKTRR:      MOV     AL,BH
042D 5E                              POP     SI
042E 5A                              POP     DX
042F 59                              POP     CX
0430 5B                              POP     BX
0431 C3                              RET
                         ;
                         ;
                         ;=================================
                         ;===                           ===
                         ;=== TRANSLATE 3D TO 2D ===========
                         ;===                           ===
                         ;=================================
                         ;
                         TRN_3D2D:
0432 BE0000                          MOV     SI,0
0435 BF0000                          MOV     DI,0
0438 A1091C                          MOV     AX,WORD PTR NO+2
043B 2B06031C                        SUB     AX,WORD PTR EP+2
043F BB0D1C                          MOV     BX,OFFSET PTB
0442 B500                            MOV     CH,0
0444 8A0E4F1C                        MOV     CL,PMAX
0448 FEC1                            INC     CL
044A 51                  LP2:        PUSH    CX
044B 50                              PUSH    AX
044C 8B16051C                        MOV     DX,WORD PTR EP+4
0450 2B5004                          SUB     DX,[BX+SI]+4
0453 F7EA                            IMUL    DX
0455 8B4802                          MOV     CX,[BX+SI]+2
0458 2B0E031C                        SUB     CX,WORD PTR EP+2
045C F7F9                            IDIV    CX
045E 03060B1C                        ADD     AX,NO+4
0462 2B06051C                        SUB     AX,EP+4
0466 89850314                        MOV     TRANS_TBL+2[DI],AX
                         ;
046A 8B10                            MOV     DX,[BX+SI]
```

```
046C 2B16011C                    SUB     DX,EP
                          ;
0470 58                          POP     AX
0471 F7EA                        IMUL    DX
0473 8B4802                      MOV     CX,[BX+SI]+2
0476 2B0E031C                    SUB     CX,EP+2
047A F7F9                        IDIV    CX
047C 0306011C                    ADD     AX,EP
0480 2B06071C                    SUB     AX,NO
0484 89850114                    MOV     TRANS_TBL[DI],AX
0488 83C606                      ADD     SI,6H
048B 83C704                      ADD     DI,4H
                          ;
048E 59                          POP     CX
048F E2B9          044A          LOOP    LP2

                          ;
                          ;==================================
                          ;=== INITIALIZE ===================
                          ;==================================
0491 CDA0                        INT     GINIT
                          ;==================================
                          ;=== SCREEN COMAND ================
                          ;==================================
0493 BB4D01                      MOV     BX,OFFSET SCREEN
0496 CDA1                        INT     GSCREEN
                          ;==================================
                          ;=== CLS COMMAND ==================
                          ;==================================
0498 CDA5                        INT     GCLS
                          ;==================================
                          ;=== PLOT =========================
                          ;==================================
049A 8A0E4F1C                    MOV     CL,PMAX
049E 880E5101                    MOV     POINT_W,CL
04A2 E85700        04FC PLOT:    CALL    POINT
04A5 FE0E5101                    DEC     POINT_W
04A9 79F7          04A2          JNS     PLOT
04AB C3                          RET
                          ;==================================
                          ;===                            ===
                          ;=== SYSTEM RESET ================
                          ;===                            ===
                          ;==================================
                          REBOOT:
04AC B100                        MOV     CL,0
04AE B200                        MOV     DL,0
04B0 CDE0                        INT     BDOS

                          ;==================================
                          ;===                            ===
                          ;=== STOP INTERRUPT ==============
                          ;===                            ===
                          ;==================================
                          STOP_KEY:
04B2 1E                          PUSH    DS
04B3 2E8E1EF604                  MOV     DS,CS:DS_BACKUP
04B8 C6064C0101                  MOV     STOP_FLAG,1
04BD 1F                          POP     DS
04BE CF                          IRET
                          ;==================================
                          ;===                            ===
                          ;=== CHECK STOP =================
                          ;===                            ===
                          ;==================================
                          CHECK_STOP:
04BF 1E                          PUSH    DS
04C0 2E8E1EF604                  MOV     DS,CS:DS_BACKUP
04C5 803E4C0100                  CMP     STOP_FLAG,0
04CA 7403          04CF          JZ      CONTINUE
04CC E90200        04D1          JMP     ABORT
                          CONTINUE:
04CF 1F                          POP     DS
04D0 CF                          IRET
                          ABORT:
```

```
04D1 FA                 CLI
04D2 CDA5               INT     0A5H
04D4 33C0               XOR     AX,AX
04D6 8EC0               MOV     ES,AX
04D8 BE0001             MOV     SI,OFFSET VECT_BACKUP
04DB BF8002             MOV     DI,0A0H*4
04DE B92000             MOV     CX,32
04E1 FC                 CLD
04E2 F3A5               REP     MOVSW
04E4 BF3803             MOV     DI,0CEH*4
04E7 A5                 MOVSW
04E8 A5                 MOVSW
04E9 BF1403             MOV     DI,0C5H*4
04EC A5                 MOVSW
04ED A5                 MOVSW
04EE BF1800             MOV     DI,6*4
04F1 A5                 MOVSW
04F2 A5                 MOVSW
04F3 FB                 STI
04F4 7AB6      04AC     JP      REBOOT
                ;==================================
                ;=== SEGMENT ADDRESS STORE AREA ===
                ;==================================
                ;
04F6 0000       DS_BACKUP       DW      0
04F8 0000       SS_BACKUP       DW      0
04FA 0000       SP_BACKUP       DW      0
                ;
                ;==================================
                ;=== POINT ========================
                ;==================================
                POINT:  PUSH    CX
04FC 51
04FD BE5101             MOV     SI,OFFSET POINT_W
0500 8A04               MOV     AL,BYTE PTR[SI]
0502 32E4               XOR     AH,AH
0504 BB0400             MOV     BX,4H
0507 F7E3               MUL     BX
0509 96                 XCHG    AX,SI
050A BB0114             MOV     BX,OFFSET TRANS_TBL
050D BF5501             MOV     DI,OFFSET LIO_W
0510 8B10               MOV     DX,WORD PTR [BX+SI]
0512 8915               MOV     WORD PTR [DI],DX
0514 8B5002             MOV     DX,WORD PTR [BX+SI]+2
0517 895502             MOV     WORD PTR [DI]+2,DX
051A 96                 XCHG    AX,SI
051B 32E4               XOR     AH,AH
051D 8A4401             MOV     AL,BYTE PTR [SI]+1
0520 894504             MOV     WORD PTR [DI]+4,AX
                ;
0523 B101               MOV     CL,1-HDV
0525 D3F8               SAR     AX,CL
                ;
0527 894506             MOV     WORD PTR [DI]+6,AX
052A 8A4402             MOV     AL,BYTE PTR [SI]+2
052D 884508             MOV     BYTE PTR [DI]+8,AL
0530 C6450900           MOV     BYTE PTR [DI]+9,0H
0534 8BDF               MOV     BX,DI
0536 56                 PUSH    SI
0537 53                 PUSH    BX
0538 CDA8               INT     0A8H
053A 5B                 POP     BX
053B 5E                 POP     SI
053C 8A4403             MOV     AL,BYTE PTR [SI]+3
053F 3C7F               CMP     AL,07FH
0541 7505      0548     JNE     RET_POINT
0543 884704             MOV     BYTE PTR [BX]+4,AL
0546 CDA6               INT     0A6H
                RET_POINT:
0548 59                 POP     CX
0549 C3                 RET
                ;==================================
                ;=== LINE =========================
                ;==================================
                LINE:
054A 50                 PUSH    AX
054B 53                 PUSH    BX
054C 51                 PUSH    CX
```

```
054D 52                    PUSH   DX
054E 57                    PUSH   DI
054F 56                    PUSH   SI
0550 52                    PUSH   DX
0551 50                    PUSH   AX
0552 86C4                  XCHG   AL,AH
0554 E82D00        0584    CALL   LINE_1
0557 BF5501                MOV    DI,OFFSET LIO_W
055A 8905                  MOV    WORD PTR [DI],AX
055C 8B4002                MOV    AX,WORD PTR [BX+SI]+2
055F 894502                MOV    WORD PTR [DI]+2,AX
0562 58                    POP    AX
0563 E81E00        0584    CALL   LINE_1
0566 894504                MOV    WORD PTR [DI]+4,AX
0569 8B4002                MOV    AX,WORD PTR [BX+SI]+2
056C 894506                MOV    WORD PTR [DI]+6,AX
056F 58                    POP    AX
0570 32E4                  XOR    AH,AH
0572 894508                MOV    WORD PTR [DI]+08H,AX
0575 C6450A00              MOV    BYTE PTR [DI]+0AH,0H
0579 8BDF                  MOV    BX,DI
057B CDA7                  INT    0A7H
057D 5E                    POP    SI
057E 5F                    POP    DI
057F 5A                    POP    DX
0580 59                    POP    CX
0581 5B                    POP    BX
0582 58                    POP    AX
0583 C3                    RET
                   ;
0584 B404          LINE_1: MOV    AH,04H
0586 F6E4                  MUL    AH
0588 8BF0                  MOV    SI,AX
058A BB0114                MOV    BX,OFFSET TRANS_TBL
058D 8B00                  MOV    AX,WORD PTR [BX+SI]
058F C3                    RET
                   ;
                   ;
                   ;=====================================
                   ;===                               ===
                   ;=== DATA SEGMENT ==================
                   ;===                               ===
                   ;=====================================
                   ;
                           DSEG
                           ORG    100H

 F990              LIO_BASE       EQU    0F990H
 0000              HDV            EQU    0         ;1=HIGH RESOLUTION,
                                                   ;0=LOW RESOLUTION

0100               VECT_BACKUP    RW     38
014C 00            STOP_FLAG      DB     0
                   ;
                   ;PARAMETER LIST
                   ;
014D 00            SCREEN         DB     HDV*3
014E 00                           DB     0
014F 00                           DB     0
0150 01                           DB     1
                   ;
0151 00            POINT_W        DB     0
0152 05                           DB     5
0153 03                           DB     3
0154 7F                           DB     7FH

                   ;==================================
                   ;=== LIO WORK AREA ================
                   ;==================================
0155               LIO_W          RB     300H
0455                              RB     0
                   ;==================================
                   ;=== GRAPH LIO SYSTEM AREA ========
                   ;==================================
                                  ORG    620H
0620                              RB     0DE0H
1400 00                           DB     00H
                   ;
                   ;
```

```
 1401                       TRANS_TBL    RW    1024
=                           INCLUDE      B:TABLE.DAT
=                           ;
=                           ; EP : EYE POINT
=                           ;
=1C01 00000000FA00          EP           DW    0,       0,      250
=                           ;
=                           ; NO : NORMALIZED ORIGIN
=                           ;
=1C07 C0FEC8009001          NO           DW    -320,    200,    400
=                           ;
=                           ; PTB : POINT TABLE DATA
=                           ;
=1C0D 8C008A02B400          PTB          DW    140,     650,    180  ;0
=1C13 A2FE8A02B400                       DW    -350,    650,    180  ;1
=1C19 1801E402C800                       DW    280,     740,    200  ;2
=1C1F 40010C03BE00                       DW    320,     780,    190  ;3
=1C25 04018403C800                       DW    260,     900,    200  ;4
=1C2B ECFF9803BE00                       DW    -20,     920,    190  ;5
=1C31 640052037C01                       DW    100,     850,    380  ;6
=1C37 E2FF0C03A401                       DW    -30,     780,    420  ;7
=1C3D C80002039001                       DW    200,     770,    400  ;8
=1C43 F0002003A401                       DW    240,     800,    420  ;9
=1C49 A0002003F401                       DW    160,     800,    500  ;10
=                           ;
=                           ; PMAX : (POINT NUMBER - 1)
=1C4F 0A                    PMAX         DB    10
=                           ;

 1C50 00                                 DB    0
                            ;
 1C51 06                    NUM6    DB   6
 1C52 0800                  NUM8    DW   8
 1C54 0002                  NUM512  DW   512
                            ;
 1C56                       LTB     RS   1024
                            ;
 2056                       LP      RW   1
 2058                       LS      RW   1
 205A                       FNXB0   RB   1
 205B                       FNXB1   RB   1
 205C                       MD      RB   1
 205D 00                            DB   0
                            ;
                            ;==================================
                            ;=== STACK AREA ===================
                            ;==================================
 205E                               RW        1024
 285E 00                    STACK_TOP  DB     0
 285F 00                            DB   0
                            ;
                                    END

END OF ASSEMBLY.  NUMBER OF ERRORS:   0.  USE FACTOR:  3%
```

```
0000 VARIABLES
04F6 DSBACKUP   1C01 EP          205A FNXB0       205B FNXB1       0155 LIOW
2056 LP         2058 LS          1C56 LTB         205C MD          1C07 NO
1C54 NUM512     1C51 NUM6        1C52 NUM8        1C4F PMAX        0151 POINTW
1C0D PTB        014D SCREEN      04FA SPBACKUP    04F8 SSBACKUP    285E STACKTOP
014C STOPFLAG   1401 TRANSTBL    0100 VECTBACKUP

0000 NUMBERS
00E0 BDOS       00A8 GCIRCLE     00A5 GCLS        00A3 GCOLOR1     00A4 GCOLOR2
00CE GCOPY      00AB GGET        00A0 GINIT       00A7 GLINE       00A9 GPAINT1
00AA GPAINT2    00AF GPOINT2     00A6 GPSET       00AC GPUT1       00AD GPUT2
00AE GROLL      00A1 GSCREEN     00A2 GVIEW       0000 HDV         FFFF INF
F990 LIOBASE    7FFF MAXVAL      8000 MINVAL      FFFE NIL

0000 LABELS
04D1 ABORT      04BF CHECKSTOP   0147 CKLIN1      0182 CKLNR1      0181 CKLNR2
04CF CONTINUE   00D5 DUMM        021C FNARC       01E3 FNDT        0272 FNX
01B2 GETPTB     00C3 ILIN        00C9 ILIN1       00EA ILIN2       054A LINE
0584 LINE1      044A LP2         0050 LPSETV      0078 MAIN        0185 MKLIN1
0396 MKTR1      03A4 MKTR2       03EF MKTR3       0400 MKTR4       0404 MKTR5
0417 MKTR6      0373 MKTRI       042B MKTRR       0370 MKTRR1      0086 OLIN1
008B OLIN2      0098 OLIN3       00A5 OLIN4       00A9 OLIN5       00AE OLIN6
00B7 OLIN7      04A2 PLOT        04FC POINT       0107 PSORT       010C PSORT1
0114 PSORT2     011C PSORT3      0129 PSORT4      013E PSORT5      04AC REBOOT
02F0 REGLN      030B REGLN1      0548 RETPOINT    0337 SCHLN1      0343 SCHLN2
0352 SCHLN3     035E SCHLN4      036C SCHLNR      01C9 SQRT        01D2 SQRT1
01DE SQRT2      0331 SRCHLN      04B2 STOPKEY     0432 TRN3D2D
```

リスト2

```
;
; EP : EYE POINT
;
EP              DW      0,      0,      250
;
; NO : NORMALIZED ORIGIN
;
NO              DW      -320,   200,    400
;
; PTB : POINT TABLE DATA
;
PTB             DW      140,    650,    180     ;0
                DW      -350,   650,    180     ;1
                DW      280,    740,    200     ;2
                DW      320,    780,    190     ;3
                DW      260,    900,    200     ;4
                DW      -20,    920,    190     ;5
                DW      100,    850,    380     ;6
                DW      -30,    780,    420     ;7
                DW      200,    770,    400     ;8
                DW      240,    800,    420     ;9
                DW      160,    800,    500     ;10
;
; PMAX : (POINT NUMBER - 1)
PMAX            DB      10
;
```

付　録

〔用語解説〕

（アルファベット順）

ＢＩＯＳ：

Basic Input Output Subroutine 第２章を参照のこと

ＣＲＴ：

Cathode Ray Tube モニタディスプレイのこと

ＧＤＣ：

グラフィック・ディスプレイ・コントローラ　１．３参照

ＩＯアドレス：

８０８６ＣＰＵのもつＩＮ，ＯＵＴの機械語実行時に使われる．メモリアドレスとは，独立に存在する．ＩＯアドレスは一般に８ビットで表わされ，００Ｈ～０ＦＦＨである．

ＬＩＯ：

Logical Input Output 第３章を参照のこと

ＬＰ：

Last Referenced Point. 最後に操作した座標をいう．相対座標指定（ＳＴＥＰ）は，このＬＰからの相対値である．ＬＰはワールド座標を評価するたびに更新されるので，"(Wx1,Wy1)-STEP(Wx2,Wy2)" の第２項は第１項からの相対値となる．各ＢＡＳＩＣコマンドのＬＰに与える影響については４．０を参照のこと．画面上ではＬＰは１点だが，表現方法で２種類のＬＰが存在する．つまり，ワールド座標上か，スクリーン座標上かである．ワークエリアには，この２種類が格納されており，本書ではそれぞれワールドＬＰ，スクリーンＬＰと呼んでいる．

ＶＲＡＭ：

ビデオＲＡＭを参照のこと

ＶＳＹＮＣ：

垂直同期信号期間．１．３を参照のこと

（あいうえお順）

アトリビュート：

属性．テキスト画面で，色をつけたり，点滅，反転させたりする情報．１．４を参照のこと

解像度：

１画面に何ドット表示できるかをいう．または分解能．ＰＣ－９８００シリーズでは最高６４０×４００．

グラフィック画面：

ＰＣ－９８００シリーズでは，グラフィックを表示する画面と文字を表示する画面とを用意し，前者をグラフィック画面，後者をテキスト画面と呼んでいる．グラフィック画面では，画面の１ドットをメモリの１ビットに対応させている．テキスト画面では，文字のコードが画面には書かれていて，表示のときにキャラクタジェネレータと呼ばれるフォントパターンのＲＯＭを参照し，リアルタイムでドットパターンに変換されて表示される．１．１参照

座標系：

ＰＣ－９８００シリーズのN88－ＢＡＳＩＣには，ワールド座標系，スクリーン座標系，オリジナルスクリーン座標系の３つの座標系がある．ワールド座標系とは，仮想画面とも呼ばれ，実際の画面のドット数に関係なく自分で定義できるもので，それに対して，スクリーン座標系，オリジナルスクリーン座標系は，物理的に決まるもので，画面の１ドットが１単位となっている．オリジナルスクリーン座標系では，常に画面の左上の角を（０，０）とするのに対してスクリーン座標系では，ビューポートの左上の角を（０．０）とする．これらの概念はＢＩＯＳ，ＬＩＯにはなくＢＡＳＩＣだけに存在する．ＢＩＯＳ，ＬＩＯ等で扱う座標はすべてこのオリジナルスクリーン座標に相当する．

スクリーンＬＰ：

ＬＰを参照のこと

セミグラフィック：

ＰＣ－９８００シリーズでは，ＰＣ－８０００シリーズ，ＰＣ－８８００シリーズと互換性を持たせるため，ＰＣ－８００１でのグラフィック機能を実現した．これをセミグラフィックと呼ぶ．これはテキスト画面でグラフィックを行うもので，１文字分の面積を８分割してそれぞれテキスト画面のメモリの１ビットに対応させる．セミグラフィックでは最大１６０×１００のグラフィックとなる．

テキスト画面：

グラフィック画面を参照のこと

ドット：

画面上の１点.

パレットレジスタ：

１．２を参照のこと

バンク切り換え：

１．６を参照のこと

ビデオＲＡＭ：

メモリの内容が画面に出力される特別の機能を持ったＲＡＭ．ＰＣ－９８００シリーズではテキスト用とグラフィック用にそれぞれ用意されている．１．１を参照のこと

不揮発性メモリ：

電源を消しても内容が消えないように電池でバックアップされているメモリ.

プレーン：

グラフィックＲＡＭ９６Ｋバイトを３つに分け，その１つをプレーンと呼ぶ.各プレーンは単色の１画面を構成し，３つを合成することでカラー８色を表示できる.

フリップフロップ：

１か０かどちらかの状態を保持しておくもの，１ビットのメモリともいえる.

ブリンキング：

点滅すること.

マルチプレクサ：

複数の入力から１つを選択して出力するハードウエア.

ワールドＬＰ：

ＬＰを参照のこと

著　者　井上　智博　　TOMOHIRO　INOUE

発行者　牧谷秀昭
発行所　秀和システムトレーディング株式会社
郵便番号１０７　東京都港区南青山２－１９－５　関原ビル

SHUWA SYSTEM TRADING CO., LTD.
SEKIHARA BLDG,
2-19-5 MINAMIAOYAMA, MINATO-KU, TOKYO, 107 JAPAN

印刷所　東京スガキ印刷株式会社

発行日　１９８４年１２月１８日

秀和システムトレーディング株式会社　　定価　3,200円